Министерство образования и науки Российской Федерации

Псковский государственный университет

Полетаев И.А., Полетаева О.А.

ПРОГРАММИРОВАНИЕ НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ ПАСКАЛЬ

Методические указания по выполнению лабораторных работ для студентов очной формы обучения направлений подготовки
09.03.01 «Информатика и вычислительная техника»,
09.03.02 «Информационные системы и технологии»,
09.03.04 «Программная инженерия»,
13.03.02 «Электроэнергетика и электротехника»

Псков
2018

ББК 32.973.26-018.1
УДК 681.3.06
П49

Рекомендовано к изданию научно-методическим советом
Псковского государственного университета

Рецензент:
- М.И. Швец, зам. Директора ГБУ ПО «Региональный центр информационных технологий.

Полетаев И.А., Полетаева О.А. Программирование на алгоритмическом языке Паскаль. Методические указания по выполнению лабораторных работ для студентов очной формы обучения. – Псков, Изд-во ПсковГУ, 2018. – 112 с.

В методических указаниях «Программирование на алгоритмическом языке Паскаль» изложены основные методы работы в среде программирования Турбо Паскаль с использованием IBM-совместимых персональных компьютеров; описание тринадцати лабораторных работ. В каждом задании приведено по 30 вариантов, так же приведены правила оформления лабораторных работ.

Методические указания предназначены для студентов очной формы обучения направлений подготовки 09.03.01 «Информатика и вычислительная техника», 09.03.02 «Информационные системы и технологии», 09.03.04 «Программная инженерия», 13.03.02 «Электроэнергетика и электротехника», а так же могут быть использованы студентами других специальностей для выполнения лабораторных работ по аналогичным курсам.

Табл. 15. Ил. 30. Библиогр. 8 назв.

Содержание

Введение

Общие положения

Решение задачи на ЭВМ с составлением программы состоит из четырех этапов:

1. Постановка задачи.
2. Составление алгоритма.
3. Составление программы.
4. Ввод и отладка программы.

То есть, прежде чем приступать к непосредственному составлению программы или написанию последовательности операторов языка, необходимо отчетливо представить себе ход процесса вычислений, ту последовательность действий, которую должна реализовать программа. Первый этап не поддается строгой формализации и может быть достаточно сложным для больших задач, но в лабораторных работах постановка задачи приводится в самом задании.

Строгое представление последовательности действий, то есть алгоритм, наиболее удобно изображать графически с помощью блок-схем или граф-схем, хотя его можно представлять и другими способами, например словесно. На этапе обучения программированию использование детализированных блок-схем является обязательным.

Составление программы выполняется на каком-либо языке программирования. В данном курсе используется язык высокого уровня Паскаль. При составлении программы необходимо строго придерживаться правил записи программы, которые изложены в литературе, например, приведенной в списке в конце методических указаний.

Для ввода и отладки программы используется система программирования Турбо Паскаль, или Free Pascal, или Pascal ABC: краткие сведения о первой из них приводятся далее. Этот этап включает и тестирование программы, то есть проверку ее работоспособности при самых разнообразных условиях эксплуатации и вводимых данных.

При профессиональном программировании существует и пятый этап – сопровождение программы. Он заключается в исправлении замеченных в ходе работы с программой ошибок и изменениях по улучшению эксплуатационных свойств про-

граммы. Для больших программных комплексов (например, операционных систем) это один из самых трудоемких этапов, наряду с тестированием программы.

Основные сведения об алгоритмах

Алгоритм – это формальное предписание, однозначно определяющее содержание и последовательность операций, переводящих совокупность исходных данных в искомый результат – решение задачи.

Иначе говоря, алгоритм – это набор понятных исполнителю инструкций (команд), точное выполнение которых приводит к достижению требуемого результата.

Алгоритмы имеют определенную форму записи. При составлении сложных алгоритмов, которые используются в программировании, чаще всего используется графическая форма записи. На неё введены соответствующие стандарты: «ГОСТ 19.003-80. Схемы алгоритмов и программ. Обозначения условные графические» и «ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения».

Линии, указывающие последовательность перехода от блока к блоку, называемые линиями потока, и линии контуров блоков должны иметь одинаковую толщину. Основное направление потока информации идет сверху вниз и слева направо, здесь стрелки на линиях можно не указывать. В остальных случаях наличие стрелок обязательно. Отношение ширины и высоты блока строго регламентировано и обычно составляет 3 к 2 (***b*** к ***a***, если не указано отдельно, т.к. некоторые блоки имеют половинную высоту).

Приведем самые основные блоки, которые будут использоваться в лабораторных работах.

1. Вычислительный блок, ***Процесс***, изображаемый прямоугольником с входящей и исходящей стрелками (рис.1). В блоке указывается (с различной степенью детализации) последовательность реализуемых действий.



Рис. 1. Блок Процесс.

2. Блоки ввода и вывода информации, ***Ввод-вывод***, изображаются параллелограммом с входящей и исходящей стрелками. Это относится к любым носителям информации (рис.2). В блоке указываются вводимые или выводимые данные.

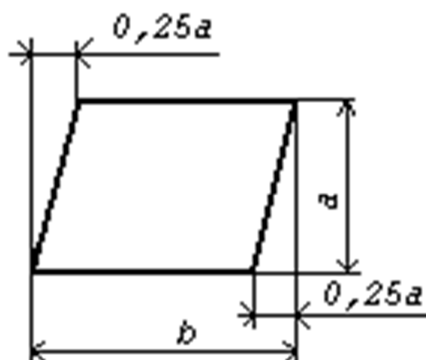


Рис. 2. Блок *Ввод-вывод.*

3. Блок начала программы, ***Пуск***, представляет собой овал с выходящей из него линией. В овале может быть приведена вспомогательная или поясняющая информация. Аналогично блок окончания программы, ***Останов,*** иначе называемый ***Терминатор*** (рис.3).

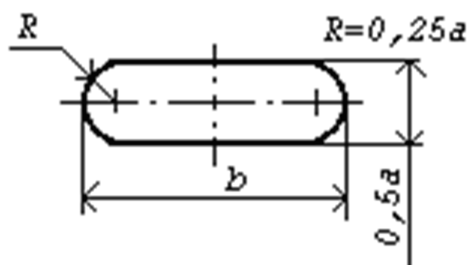


Рис. 3. Блок *Пуск-останов.*

4. Логический блок, ***Решение***, изображаемый в виде ромба с одной входящей и двумя или несколькими выходящими

стрелками (рис.4). Внутри ромба помещается текст логического вопроса, допускающего или двоичный ответ (да/нет), или несколько вариантов выбора. В любом случае над стрелками пишутся условия прохождения по этой ветви.

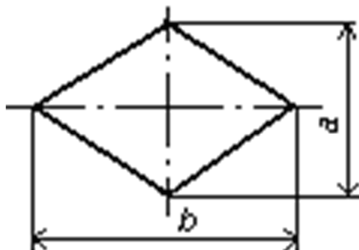


Рис. 4. Блок *Решение*.

5. Специально для отображения циклических структур введен блок заголовка цикла, ***Модификация***, после которого идут блоки внутрициклповых операций (рис.5). С последнего блока линия потока должна возвращаться на заголовок цикла. Вторая линия из блока выходит по условию окончания цикла.

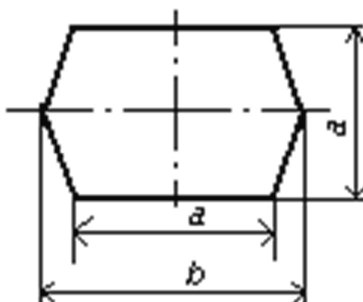


Рис. 5. Блок *Модификация*.

6. Для более наглядного представления можно использовать блок, состоящий из двух фигур, который аналогичен блоку Модификация, ***Граница цикла*** (рис.6)

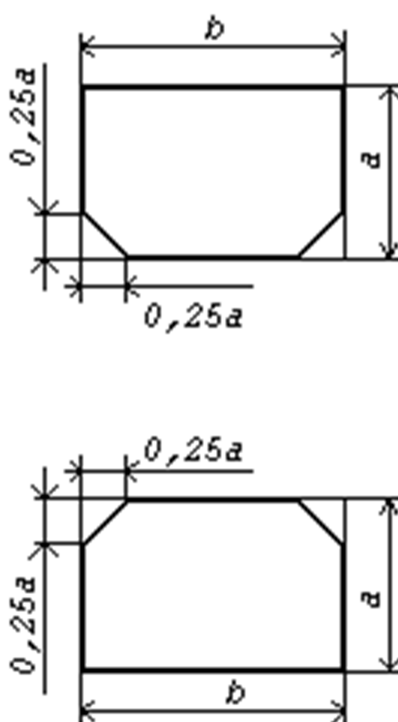


Рис. 6. Блок *Граница цикла.*

7. Если модуль или подпрограмма составлены и описаны отдельно, то используется блок ***Предопределенный процесс*** (рис.7). В нем указывается название подпрограммы или программного модуля.

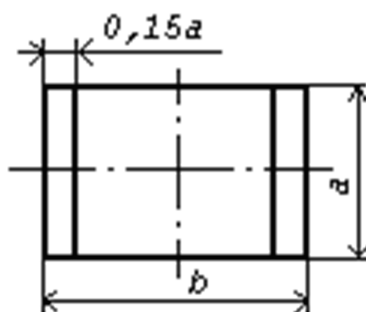


Рис. 7. Блок *Предопределенный процесс.*

8. Для пояснения отдельных блоков, их групп и линий потока используются ***комментарии*** (рис.8). Они записываются справа от блоков и соединяются с ними пунктирной линией. В комментариях может находиться любая поясняющая информация.

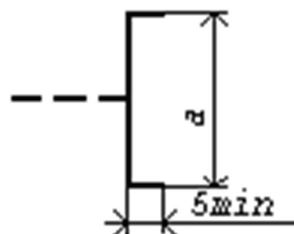


Рис. 8. Комментарий.

8. При большом количестве блоков или линий связи линии потока можно прерывать, используя ***Соединители*** (рис.9), изображаемые в виде круга. Внутри круга ставятся цифры или комбинации букв и цифр, но одинаковые в начале и в конце обрыва линии потока.

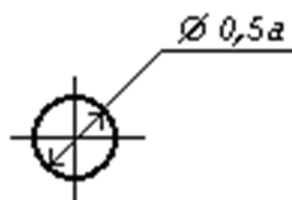


Рис. 9. Соединители.

Использование этих блоков позволяет наглядно представить алгоритм вычислений. Всего же ГОСТы устанавливают для изображения схем алгоритмов и программ 42 символа. Из них 30 – обязательных, а 12 – рекомендуемых.

Примеры блок-схем приводятся в некоторых заданиях.

После того, как составлена блок-схема алгоритма, можно писать программу на любом языке программирования, подходящем для данной области. Для научно-технических задач это такие языки, как Паскаль, Си++, Си#, Бейсик, PHP и другие. В данном курсе используется язык программирования Паскаль для IBM-совместимых компьютеров (точнее, его объектно-ориентированная реализация Object Pascal).

Язык Паскаль и интегрированные среды разработки программ

Хотя язык Паскаль является относительно старым языком программирования, созданным Никлаусом Виртом в 1968-69 годах специально для обучения студентов программированию, но с появлением персональных компьютеров он получил широкое распространение не только в образовательной сфере, но и при решении различных прикладных задач.

В 1986 году фирма Apple разработала объектное расширение языка Паскаль, получив в результате Object Pascal. Он был разработан группой Ларри Теслера, который консультировался с Никлаусом Виртом.

Турбо-Паскаль

В 1983 году появилась первая реализация инструментальной среды Турбо Паскаль, предназначенная для IBM-совместимых компьютеров. Она включала в себя оболочку (интерфейс, с помощью которого происходило общение человека с компьютером: набор и запуск программ, считывание результатов и т.д.), текстовый редактор, транслятор и отладчик. С тех пор система программирования Турбо Паскаль непрерывно совершенствовалась фирмой Borland International. Появилась развитая система подсказки, система сборки всей программы из отдельных модулей (линковщик), богатые библиотеки подпрограмм и многое другое. Комплекс программных средств, используемый программистами для разработки программного обеспечения, стали называть **интегрированной инструментальной средой разработки программ (Integrated Development Environment, IDE, иногда именуемая ИСР),** или просто средой программирования.

В 1992 году была представлена ИСР уже с использованием языка Object Pascal – Турбо-Паскаль 7.0. На этом развитие Турбо Паскаля закончилось. Окно ИСР Турбо-Паскаль представлено на рис.10.

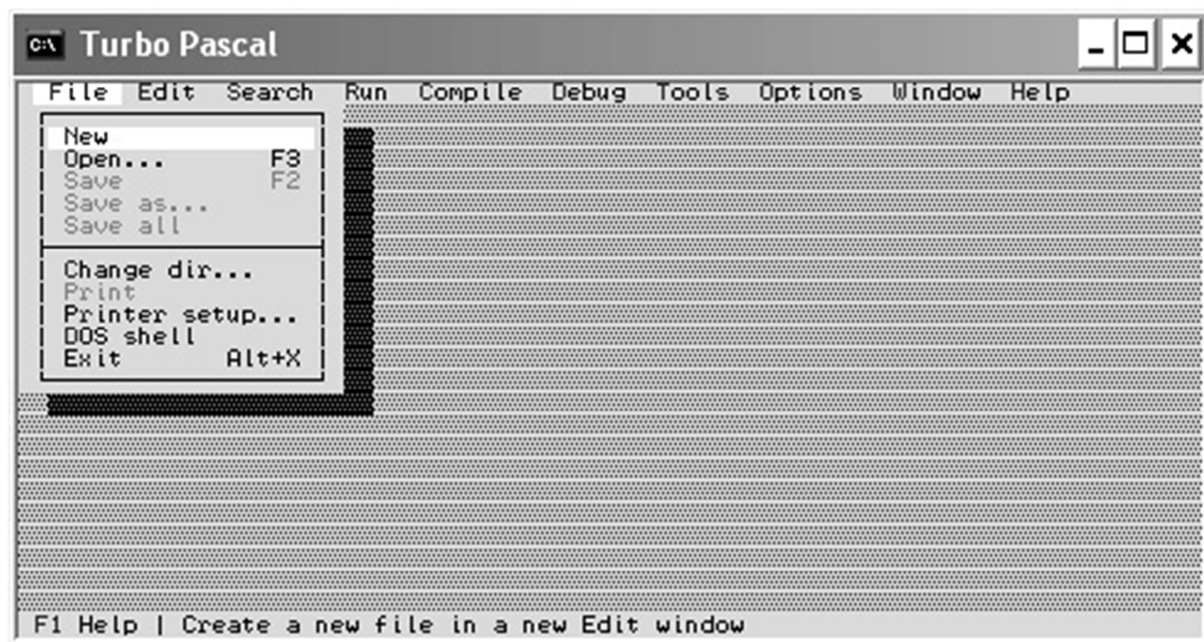


Рис. 10. Создание нового окна программы в Турбо-Паскале

Free Pascal

Свободно распространяемые компиляторы системы программирования Free Pascal реализованы во многих дистрибутивах Linux, есть свободные компиляторы и для ОС Windows XP/7/10. На основе Free Pascal создана свободная мультиплатформенная среда **Lazarus**, аналогичная среде Delphi. Free Pascal внешне очень похож на Турбо Паскаль, хотя и является мультиплатформенным.

Начальная заставка этой ИСР приведена на рис.11.

Так как Free Pascal – свободно распространяемое ПО, то его можно скачать с сайта *http://freepascal.org/* и бесплатно установить. На этом же сайте и находится вся документация по Free Pascal.



Рис. 11. Начальная заставка Free Pascal.

Pascal ABC.NET

Эта ИСР наиболее приближена к профессиональной системе Delphi, то есть разработана без оглядки на старую ОС MS DOS. Как следствие, здесь присутствует панель инструментов, закладки для оперативного переключения между программами, 2 окна: ввода и вывода результатов. Учебная система программирования Pascal ABC.NET (автор – С.С. Михалкович) представляет собой диалект стандартного языка Паскаль. Она призвана осуществить постепенный переход от простейших программ к объектно-ориентированному программированию сложных программных продуктов уже на базе Delphi.

Начало работы в этой ИСР представлено на рис.12.

Хотя и считается, что интегрированная среда разработки Pascal ABC.NET полностью совместима с языком программирования Object Pascal, но это не так. В Pascal ABC.NET достаточно много изменений, указанных в приложении Б, взятых из языков Basic, C, а главное, с платформы .NET. Поэтому, хотя поначалу кажется, что язык, используемый в Pascal ABC.NET, такой же, что и Object Pascal, но это не так.

Конечно, в некоторых случаях проще написать короткую (учебную) программу в среде Pascal ABC.NET, но все-таки лучше использовать стандартный язык программирования.

В любом случае это так же свободно распространяемое ПО, находящееся на сайте *http://pascalabc.net/*.

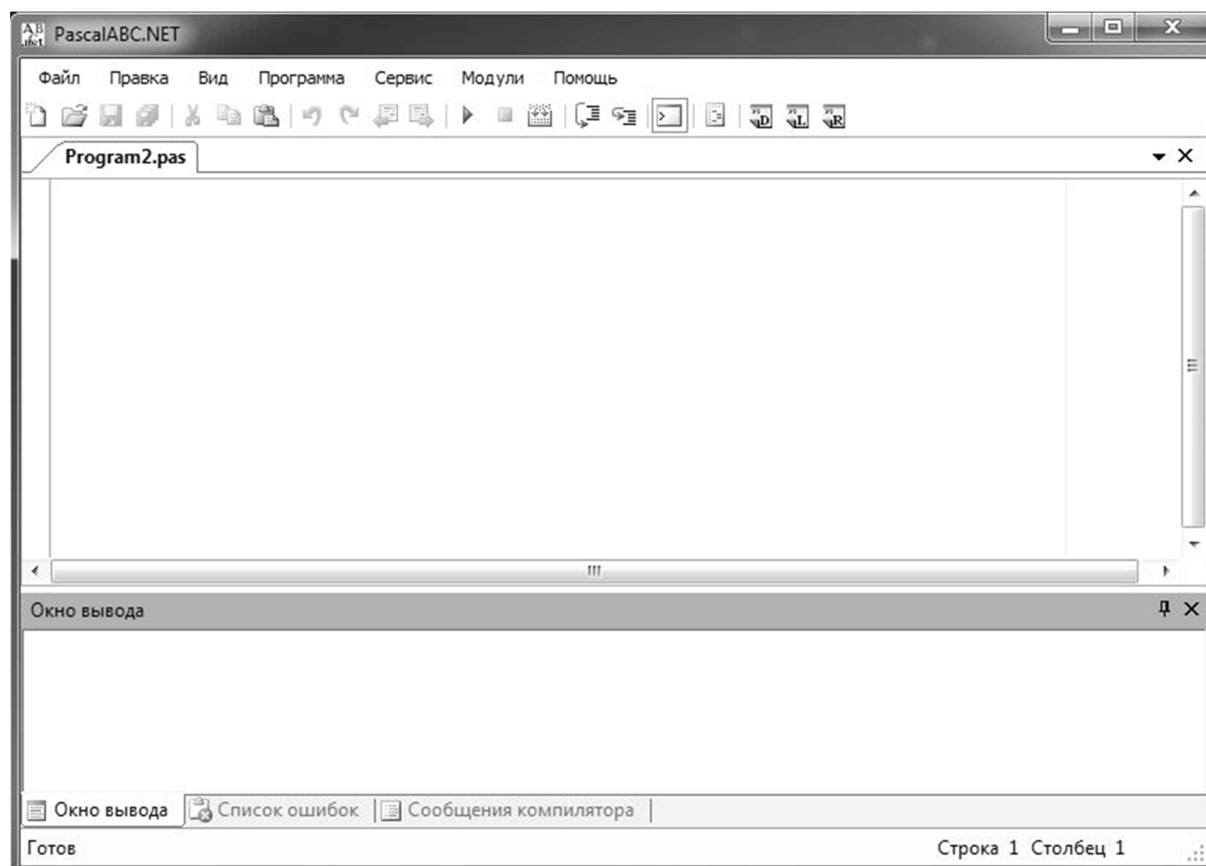


Рис. 12. Начало работы в Pascal ABC.

Delphi

Delphi является профессиональной средой программирования, хотя в его основе лежит всё тот же язык программирования Object Pascal. ИСР Delphi прошла долгий путь развития, начиная с Borland Delphi (1995 г.) для ОС Windows 3.1, затем для Windows 95 (еще 16-разрядных). Существовали версии Delphi с 2.0 до 8.0 (1996-2003 годы), Delphi 2005-2010 с соответствующими годами выпуска. Начиная с 2010 года стала выпускаются ИСР Delphi XE, затем Delphi XE2 (2011 год) и так далее.

Во введении будет кратко рассмотрена работа со средой Free Pascal. Работа с остальными ИСР очень похожа.

После того, как программа составлена, ее необходимо ввести в компьютер. Здесь мы и сталкиваемся с инструментальными пакетами программ Free Pascal или Pascal ABC. Free Pascal пакет содержит не только транслятор с языка Object Pascal, но и редактор текста, инструментальную оболочку, отладчик, описание ИСР, обширные библиотеки программ и

многое другое, например, программы примеров объектно-ориентированного программирования.

Для связи основных из этих программ в единое целое, создания удобного и наглядного интерфейса и предназначена **интегрированная инструментальная среда разработки программ, кратко ИСР**.

Вызов ИСР осуществляется посредством запуска либо с рабочего стола, либо с помощью кнопки меню «Пуск» файла fp.exe. После запуска файла fp.exe появляется основной экран ИСР, состоящий из трех частей: строки меню, рабочей зоны и строки состояния в соответствии с рис.11.

При начальном запуске ИСР в рабочей зоне открыто одно окно с номером 1 в правом верхнем углу и с заголовком *noname01.pas*. В дальнейшем, после записи программы на диск, стандартный заголовок заменится на имя программы, данное ей при записи. Если рабочая зона пустая, то создается новое окно командой *File* > *New*, как представлено на рис. 13.

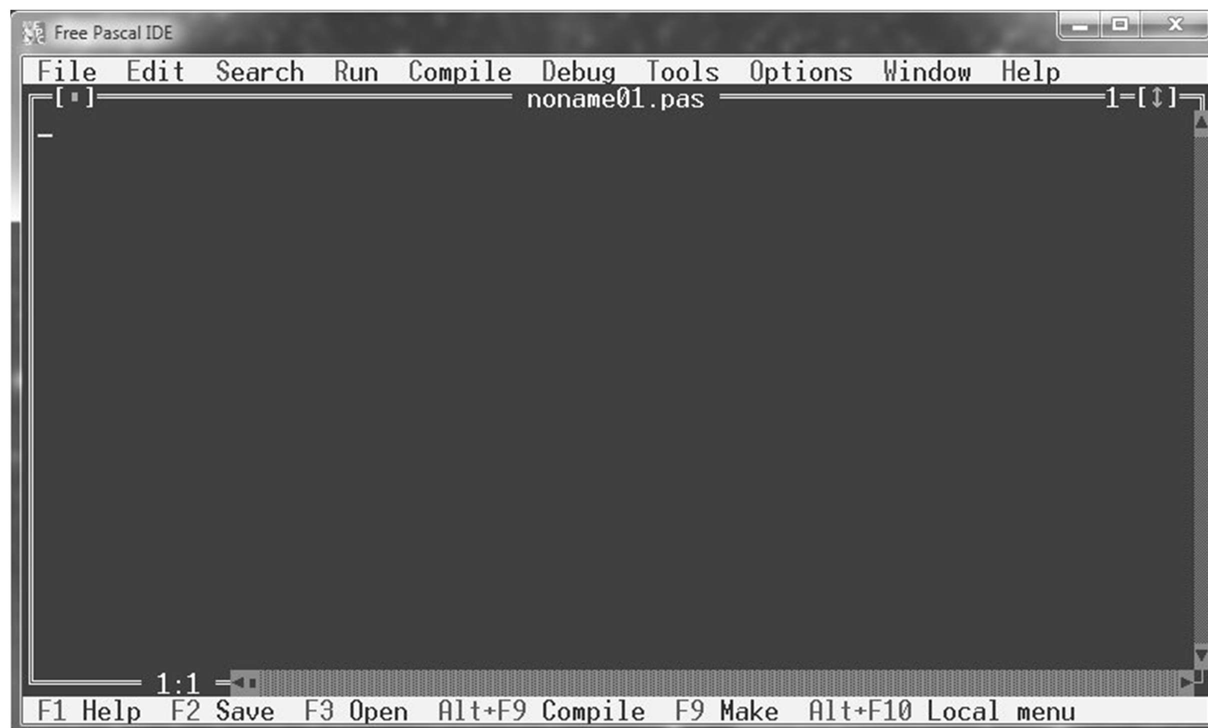


Рис. 13. Вид окна ИСР после запуска файла fp.exe в оконном режиме

Мигающий курсор указывает то место на экране, в котором будет появляться очередной символ текста. Ввод каждой новой строки заканчивается нажатием клавиши *Enter*. Компилятор не различает прописные и строчные буквы, поэтому все

равно, в каком регистре набираются латинские буквы. Так, следующие строки будут эквивалентными:

```
Program  A;
PROGRAM  a;
PrOgRaM  a;
```

Набирая текст, особое внимание необходимо обращать на точное воспроизведение всех знаков: точек, точек с запятой, апострофов, пробелов, так как компилятор очень чувствителен к мелочам подобного рода.

Строка меню активизируется клавишей F10 (если управление осуществляется с клавиатуры) и состоит из 10 пунктов, которые, в свою очередь, разворачиваются в спускающиеся подменю:

File – позволяет выполнять все основные операции с файлами: создавать новые, загружать имеющиеся, сохранять созданные и отредактированные файлы, выводить на принтер содержимое этих файлов, заканчивать сеанс с ИСР и так далее.

Edit – дает возможность выполнять основные операции редактирования текста.

Search – позволяет осуществлять поиск фрагментов текста и при необходимости производить замену найденного фрагмента новым.

Run – позволяет запускать программу, находящуюся в рабочей зоне, а так же при необходимости пошагово выполнять данную программу или ее часть. Если были внесены изменения в программу, то при запуске она автоматически заново компилируется.

Compile – возможно осуществить компиляцию программы, которая находится в рабочей зоне, без ее выполнения, чтобы проверить на наличие ошибок.

Debug – содержит команды, облегчающие процесс поиска ошибок в программе: расстановка точек остановки, визуализация окна отладки, окна регистров, окна выходных результатов и так далее.

Tools – дает возможность выполнять некоторые программы, не выходя из ИСР.

Options – здесь находятся команды, позволяющие установить необходимые для работы параметры компилятора и ИСР.

Window – позволяет выполнять все основные операции с окнами (хотя их гораздо удобнее выполнять с помощью мыши): открывать, закрывать, перемещать, изменять размер.

Help – позволяет получить имеющуюся в системе справочную информацию.

Система меню позволяет выполнять практически все команды ИСР и интегрированных программ, и выполнена в соответствии со стандартом SAA (Turbo Vision).

Строка состояния, находящаяся в нижней части экрана, в режиме редактирования демонстрирует некоторые из часто используемых операций ИСР и комбинации клавиш для их быстрого вызова, которые позволяют выполнить соответствующие операции, минуя стандартную процедуру их вызова через меню. В некоторых режимах здесь выводятся подсказки или другая справочная информация.

Предусмотрены в этой среде и всплывающие подсказки, например, на рис.14 (в рамке серого (зеленого) цвета подсказка служебного слова Program).



Рис. 14. Всплывающие подсказки в Free Pascal.

Отладка и выполнение программы

После того, как программа набрана и поставлена заключительная точка после слова *end*, ее необходимо оттранслировать, устранить ошибки, как синтаксические, так и семантические, и выполнить, то есть получить конечный результат.

После набора программы ее рекомендуется записать на диск. Более того, если текст программы достаточно объемен, лучше всего делать и промежуточные записи во избежание потери информации при сбоях компьютера или пропадании напряжения питания.

Если программа набирается заново, то есть активное окно имеет имя noname01.pas, то при нажатии клавиши *F2* выполнится команда *File/ Save as*... При этом появится диалоговое окно со списком файлов – программ из текущего раздела, с именами *Files*, как на рис.15.

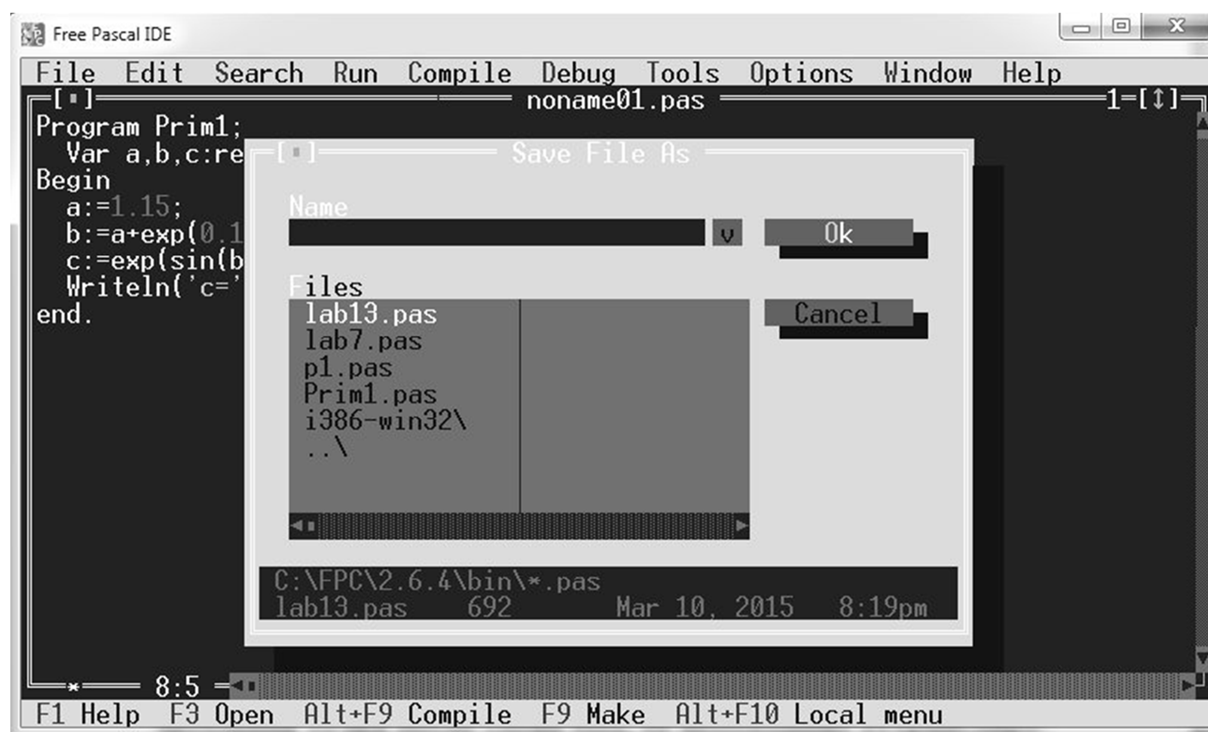


Рис. 15. Сохранение программы в Free Pascal.

В окне *Name* набирается имя файла, с которым он будет сохранен на диске. Имя автоматически будет дополнено расширением *.pas*. После записи на диск имя в текущем окне редактора сменится на заданное в поле *Name*. После дальнейшего набора программы или ее корректировки при нажатии клавиши *F2* будет выполняться команда *File/ Save*, и никаких дополнительных запросов происходить не будет.

При переходе к новой программе окно с текстом старой программы закрывают (<*Alt+F3*> или *Window / Close*, хотя это и не обязательно) и открывают новое активное окно (*File / New*). При необходимости чтения другой, ранее набранной программы, выполняется команда *File / Open...* (*F3*), в появляющемся диалоговом окне в области *Files* перемещением маркера выбирается нужный файл и нажимается клавиша *Enter*, эквивалентная кнопке диалогового окна *Open*.

После ввода программы ее можно откомпилировать с целью устранения ошибок (<*Alt+F9*>). Если программа не сохранена, то при компиляции возникнет диалоговое окно как при сохранении программы. Если компилятор обнаружил синтаксические ошибки, то появится сообщение, представленное на рис.16.

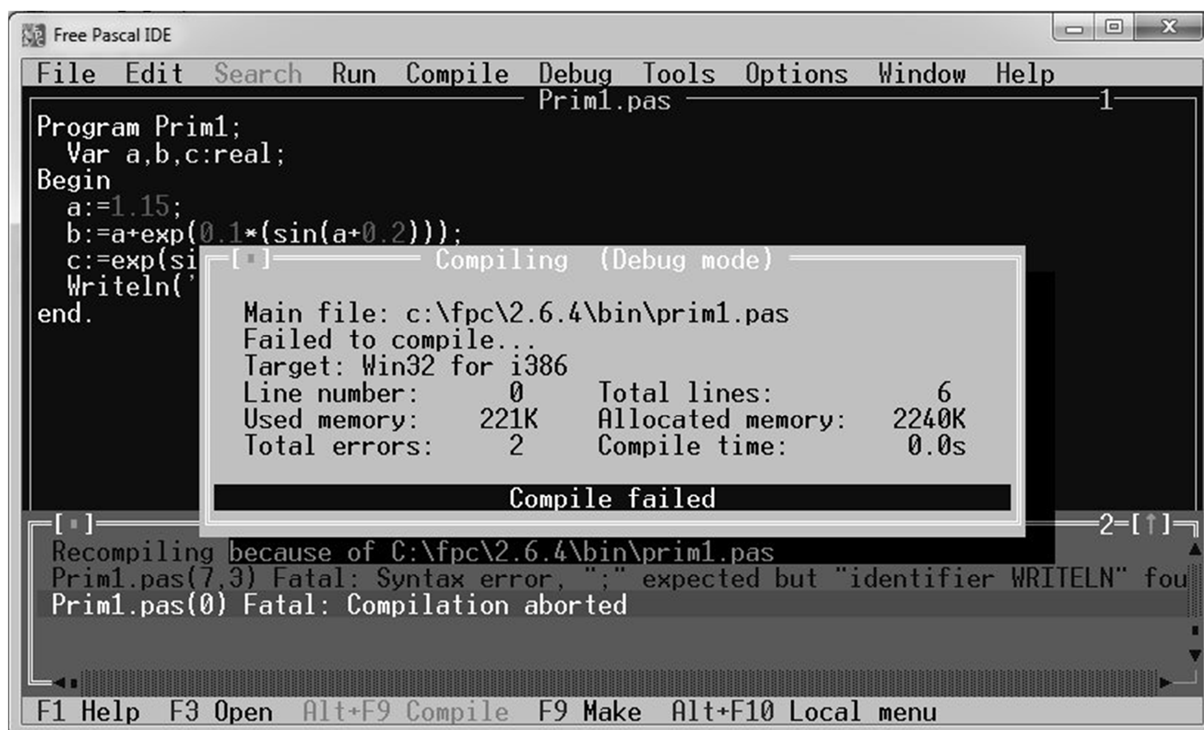


Рис. 16. Сообщение о наличии в программе синтаксических ошибок.

После того, как все ошибки устранены, появится следующее сообщение, как на рис. 17: «Компиляция выполнена: нажмите любую клавишу».

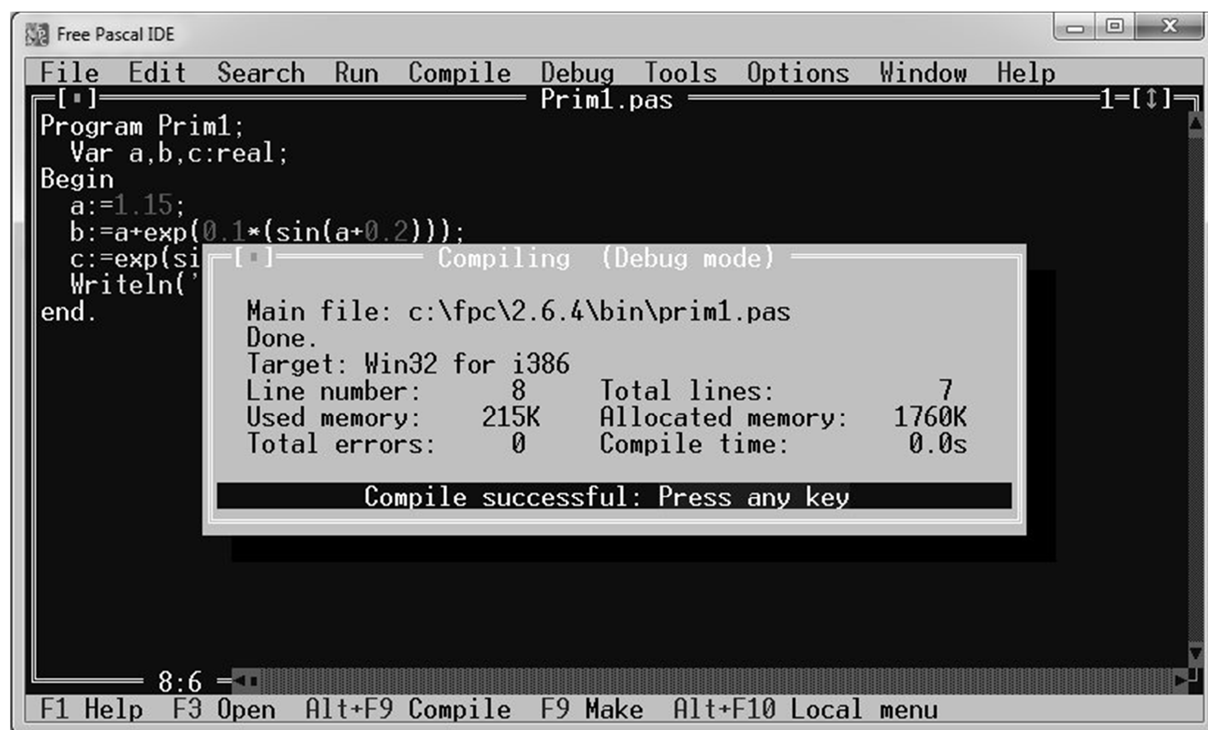


Рис. 17. Компиляция завершена успешно.

После ввода всей программы ее можно либо откомпилировать, либо сразу отдать команду на выполнение (<*Ctrl+F9*>). В последнем случае все равно выполнится предварительная компиляция, и если присутствуют синтаксические ошибки, программа выполняться не будет. Если программа не сохранена, то при компиляции возникнет диалоговое окно как при сохранении программы.

Все **ошибки программ** делятся на два больших класса: ошибки компиляции (синтаксические) и ошибки выполнения (логические или алгоритмические). О первом типе ошибок сообщает компилятор до запуска программы на выполнение с указанием типа ошибки и предполагаемого ее места. К сожалению, ошибка может быть и не там, где стоит курсор; его положение – это фактически то место, где компилятор «осознает» ошибку. Например, если имеется лишний BEGIN в программе, то компилятор не поймет этого до тех пор, пока пары BEGIN... END не будут сбалансированы. На начальном этапе программирования большинство синтаксических ошибок происходит из-за невнимательности набора программы. Даже в первой строке могут делаться ошибки, см. рис.18.

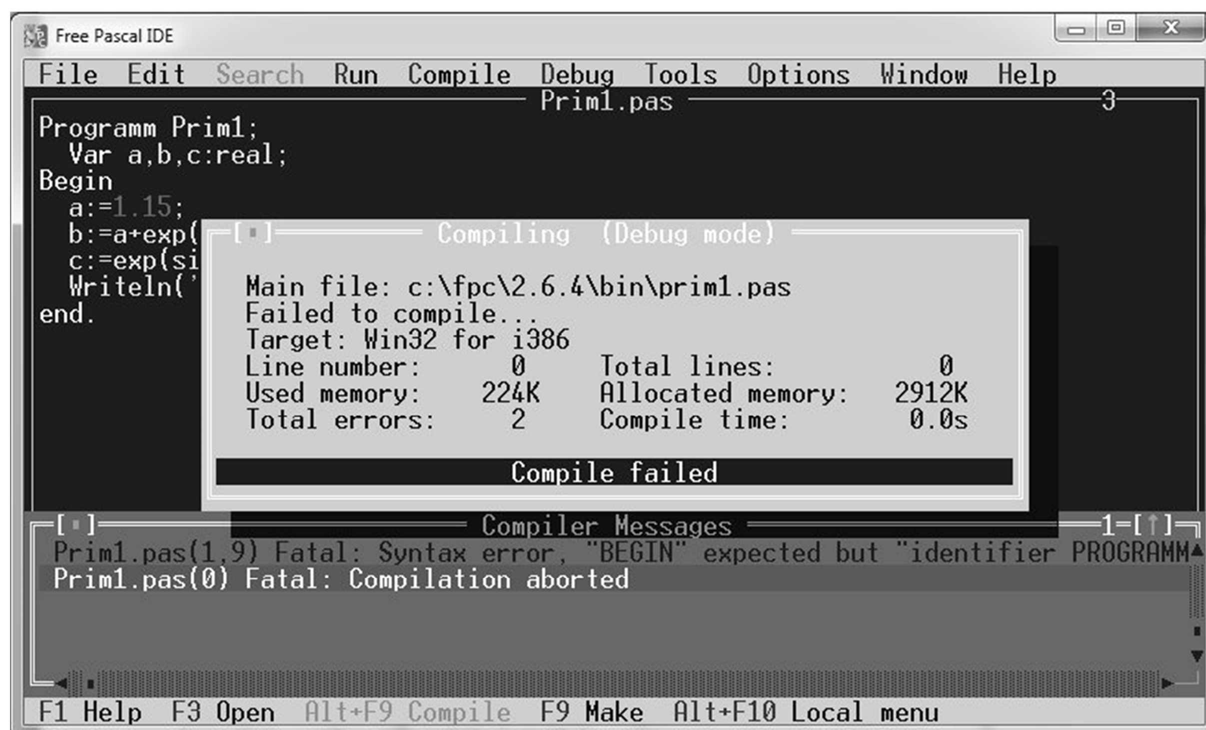


Рис. 18. Ошибка в служебном слове

В первой строке сообщений компилятора появится сообщение: «Prim1.pas(1,9) Fatal: Syntax error, “BEGIN” expected but “identifier PROGRAMM” found».

Здесь Prim1.pas – имя программы, данное ей при записи на диск.

(1,9) – место, где обнаружена ошибка (строка, столбец). Не всегда совпадает с реальным местом ошибки.

Fatal: Syntax error – синтаксическая ошибка фатальная, дальнейшее выполнение невозможно.

“BEGIN” expected but “identifier PROGRAMM” found - ожидается BEGIN, но найден идентификатор PROGRAMM. Первое обязательное служебное слово – Begin, все остальные: заголовок, разделы описаний, можно и не писать. Все конструкции языка Паскаль, которые состоят из букв и цифр и начинаются с буквы, являются идентификаторами, за исключением служебных слов и некоторых директив. Так как слово PROGRAMM к служебным не относится, то оно определено как идентификатор, с которого не должна начинаться программа.

В данном примере пояснение причины ошибки не имеет особого смысла, – ожидается оператор BEGIN, – и такое случается довольно часто. Более осмысленное толкование происходит в следующем случае:

```
Program  Prim  1;
```

...

«Prim1.pas(1,14) Fatal: Syntax error, ";"expected but "ordinal const" found» – перед порядковой константой (единицей) ожидается точка с запятой.

Смысл ошибки заключается в том, что имя программы, как и обычные идентификаторы, не должно включать пробелы, поэтому за разделителем, – пробелом, должна идти следующая конструкция языка, отделяемая от заголовка точкой с запятой.

Довольно часто позиция указывается в строке, следующей за ошибочной:

```
...
C:=2.3
A:=B*C;
```

Точка с запятой должна стоять перед А, то есть в конце предыдущей строки.

В любом случае при непонимании ошибки следует обратиться к синтаксису отмеченной конструкции языка Турбо Паскаль, либо к предыдущей.

Ошибки выполнения появляются после компиляции и запуска программы на выполнение. На экране программы выдается сообщение вида, представленного на рис.19.



Рис. 19. Сообщение с кодом завершения 201.

Ошибки выполнения искать труднее, чем синтаксические. Это и ошибки на уровне ОС, и ошибки ввода - вывода, и критические ошибки, и фатальные ошибки. Но, хотя ошибок данного класса довольно много, в данном курсе лабораторных работ чаще всего появляются только три фатальные ошибки:

Exitcode = 200 – Деление на нуль.
Exitcode = 201 – превышение времени выполнения программой.
Exitcode = 207 – переполнение при операции с плавающей точкой.

Вообще говоря, комментарии к первой и последней ошибкам не требуют дополнительных пояснений. Хотя они могут случаться не только при недопустимых делениях и вышедших из под контроля циклах операций умножения, но в данных лабораторных работах гораздо чаще встречаются при недопустимых аргументах математических функций: отрицательных значениях для логарифмов, квадратных корней, и других.

Ошибки же по времени выполнения (зацикливания компьютера) возникают либо при неверно составленном алгоритме, а соответственно, и программе, либо при выходе за границы неконтролируемых величин. Так как циклы могут использовать 2 типа операторов, то в **цикле FOR категорически**

запрещено принудительное изменение параметра цикла, так как он изменяется автоматически. **В циклах While и Repeat, наоборот обязательно надо изменять параметр цикла** принудительно, иначе он останется без изменения. В бесконечных циклах каждое последующее слагаемое **должно уменьшаться.** Пример последней ситуации приведен в примечании к лабораторной работе № 4.

ИСР Free Pascal имеет два экрана. На основном экране набираются, просматриваются и редактируются программы, отдаются команды ИСР, устанавливаются параметры работы и так далее. Но при запуске программы появляется другой экран – экран пользователя. Сюда помещаются результаты работы по программе, и он виден до тех пор, пока программа не перестанет выполняться. Так как в данном курсе лабораторных работ программы весьма просты, то они выполняются практически мгновенно, и снова появляется основной экран ИСР. **Для просмотра экрана пользователя из меню выбирается *Debug* > *User* screen или нажимается комбинация клавиш <*Alt+F5*>.** Для возврата к основному экрану можно нажать любую клавишу. Если в процессе запуска программы возникнет необходимость прервать ее работу, например в случае «зацикливания» программы, используют комбинацию клавиш <***Ctrl+Break***>.

На экране пользователя мы увидим либо сообщение об ошибке выполнения, либо результат (если пропущен оператор вывода, естественно, **вывода результата не будет**), например, как на рис.20.

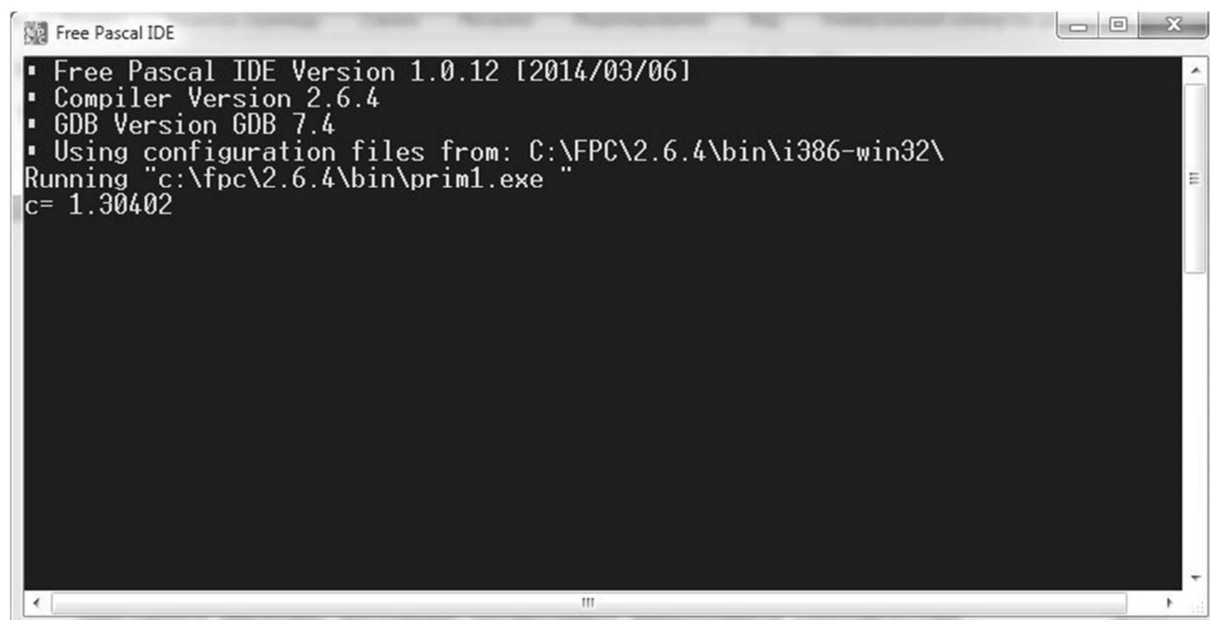


Рис. 20. Результат по нажатию клавиш <*Alt+F5*>.

Здесь нам нужна только последняя строка:
c= 1.30402

Результат может быть и неверным как из-за неправильно составленного алгоритма, так и из-за ошибок в программе, которые формально, с точки зрения транслятора, ошибками не являются. Например, при вычислении суммы по следующему фрагменту программы

```
S:=0;
For I:=1 to 10 do;
   S:=S+1/I;
Writeln (S:8:5);
```

результат вычислений будет равен 0.10000. Здесь после служебного слова DO стоит пустой оператор, поэтому именно он, то есть «ничто» будет выполняться десять раз. Затем к нулю прибавится одна десятая. Формально программа составлена правильно, и такую конструкцию иногда используют в качестве задержки по времени, но фактически десять раз должен выполняться оператор из третьей строки.

Конкретные рекомендации в таких случаях являются индивидуальными для каждой программы, существуют только общие методы тестирования. Рассмотрим два самых простейших.

Первый метод самый универсальный и может использоваться с любыми языками программирования, трансляторами

и ассемблерами. Заключается он в выводе промежуточных результатов. Внутрь цикла, или в «подозрительные» места программы временно вставляются операторы вывода изменяемых переменных. Например, программа (приведен фрагмент)

```
For  i:=1  to 15  do
  Begin
      S:=S+1/i;
       i:=i+1
   end;
```

просто «зацикливается» (если не используются директивы контроля). Но если перед i:=i+1 поставить оператор

```
Writeln (S:8:5,i:8);
```

то после запуска программы появятся два столбика бегущих чисел. Если приостановить вывод информации на экран монитора **клавишей *Pause***, то сразу будет видно, что *i* изменяется не с шагом 1, а 2. То есть параметр цикла изменяется и в операторе For, и в операторе присваивания.

Второй метод привлекает средства отладки ИСР. Предварительно установив программный счетчик на начало программы (*Run > Program reset* или <*Ctrl+F2*>), открывается окно наблюдаемых переменных (*Debug > Add Watch* или <*Ctrl+F7*>): в Watch-окне набирается имя первой контролируемой переменной, нажимается *Enter*. После этого появляется окно с именем Watches и значением этой переменной. Добавление переменных в окно выполняется необходимое количество раз. Для пооператорного выполнения программы предназначена «горячая» клавиша *F7* (*Run > Trace into*). После каждого нажатия клавиши *F7* выполняется один оператор. Если он производит изменение значения переменной, то это сразу отразится в окне наблюдения.

Таким образом, анализируя изменение значений переменных при пооператорном выполнении программы, легко найти ошибку в алгоритме, и соответственно в программе.

Порядок выполнения лабораторных работ

По каждому заданию составляется алгоритм и программа, которые помещаются в отчет.

Отчет по лабораторной работе оформляется в ученических тетрадях 12 листов или на стандартных листах формата А4, которые затем скрепляются.

Отчет должен состоять из титульного листа и описания лабораторных работ. Каждая работа представляется отдельно в отчете.

Титульный лист оформляется в соответствии с общими требованиями соответствующего факультета по оформлению титульных листов лабораторных работ. В любом случае на первом листе (для тетради – обложке) должны находиться сведения по названию предмета, номеру, названию и варианту лабораторной работы, номеру группы и ФИО учащегося.

Номер варианта задается преподавателем. Если студент не встречался с преподавателем, то номер определяется в соответствии с последними двумя цифрами шифра (номера зачетной книжки или студенческого билета) по табл. 1.

Таблица 1. Соответствие шифра (номера зачетной книжки) и номера варианта.

Номер варианта	Шифр	Номер варианта	Шифр	Номер варианта	Шифр
1	01,31,61,91	11	11,41,71	21	21,51,81
2	02,32,62,92	12	12,42,72	22	22,52,82
3	03,33,63,93	13	13,43,73	23	23,53,83
4	04,34,64,94	14	14,44,74	24	24,54,84
5	05,35,65,95	15	15,45,75	25	25,55,85
6	06,36,66,96	16	16,46,76	26	26,56,86
7	07,37,67,97	17	17,47,77	27	27,57,87
8	08,38,68,98	18	18,48,78	28	28,58,88
9	09,39,69,99	19	19,49,79	29	29,59,89
10	10,40,70,00	20	20,50,80	30	30,60,90

Каждое задание по лабораторным работам должно быть представлено четырьмя частями, выполненными рукописно или в виде распечатки электронного документа (на листах формата А4). При этом **не допускается распечатывать общие положения из данных методических указаний** для увеличения объема отчета.

1. **Вариант задания**. В кратком виде приводится задание на выполнение по конкретному варианту для возможности контроля работы, не обращаясь к данным указаниям. **Например:**

Вычислить $\sum_{i=1}^{\infty} \frac{1}{i^2 + x}$ при x =4,376 и заданной точности 10^{-4}.

2. **Блок-схема алгоритма**. Приводится рисунок с указанием фигур блоков с конкретным внутренним содержанием согласно варианту задания. Форма и соотношение размеров блоков должны соответствовать ГОСТ 19.002-80 [7].

3. **Текст программы**. Размер программы должен быть минимальным. Не следует использовать различные украшения при выводе программы: использовать графический режим, модуль CRT для очистки экрана, перемещения курсора, ожидания ввода и другие стандартные подпрограммы и модули. Если же в программе они присутствуют, то необходимо четко представлять их назначение и принципы работы, а так же отразить в алгоритме.

В задании 2 приводится два варианта текста программы.

4. **Результат вычислений**. Обычно это число, которое получается после выполнения программы в среде Турбо-Паскаль, Free Pascal или Pascal ABC, хотя это может быть и группа чисел, таблица, график. Для получения **правильного ответа** программа должна быть набрана на персональном компьютере и в ней устранены все синтаксические и логические ошибки.

Именно на этом этапе выполнения работы используется интегрированная среда разработки Турбо-Паскаль, Free Pascal или Pascal ABC основные приемы работы с Free Pascal описаны в начале методических указаний.

Этот этап при наличии компьютера выполняются на практических занятиях, в крайнем случае лабораторные работы можно выполнять самостоятельно. Тогда можно продемонстрировать умение работать в Паскаль-среде, принеся программу в электронном виде.

В любом случае должен быть представлен **отчет в бумажной форме**.

Лабораторная работа № 1. Программирование формул

Целью работы является освоение программирования алгоритмов с линейной структурой, когда решение задачи является результатом выполнения цепи вычислений, в которой очередные вычислительные действия используют в качестве исходных данных результаты вычислений на предыдущих этапах. Действия по вычислениям промежуточных и окончательных результатов описываются операторами присваивания.

Необходимо следить, чтобы порядок расположения операторов присваивания в программе от ее начала к концу соответствовал логической последовательности действий при решении поставленной задачи.

При выборе имен переменных и составлении арифметических выражений необходимо правильно устанавливать тип используемых величин (целые, вещественные и так далее). При использовании в формулах греческого алфавита можно использовать их латинские названия или буквы, сходные по начертанию. Например, символ α можно заменить на Alfa или A, ω на Omega или W.

К сожалению, в языке Паскаль имеется ограниченное количество математических функций, в ИСР Free Pascal и Pascal ABC количество функций существенно расширено (см. приложение А). Хотя при необходимости, то есть при отсутствии стандартной функции, ее выражают через другие, используя функциональные соотношения. Ниже приводятся основные математические функции, отсутствующие в языке Паскаль:

$$\arcsin x = \operatorname{arctg} \frac{x}{\sqrt{1-x^2}} \; [x^2 < 1] \;;$$

$$\arccos x = \begin{cases} \operatorname{arctg} \dfrac{\sqrt{1-x^2}}{x} & [0 < x \le 1] \\ \pi + \operatorname{arctg} \dfrac{\sqrt{1-x^2}}{x} & [-1 \le x < 0] \end{cases} \;;$$

$$\operatorname{arcctg} x = \begin{cases} \operatorname{arctg} \dfrac{1}{x} & [x > 0] \\ \pi + \operatorname{arctg} \dfrac{1}{x} & [x < 0] \end{cases} \;; \qquad \operatorname{sh} x = \frac{1}{2}\left(e^x - e^{-x}\right) \;;$$

$$\operatorname{ch} x = \frac{1}{2}\left(e^{x} + e^{-x}\right);$$

$$a^{x} = e^{x \cdot \ln a} \quad [a > 0];$$

$$\log_a x = \frac{1}{\log_x a} = \frac{\log_b x}{\log_b a} \quad [x > 0].$$

Для того чтобы использовать богатые возможности Free Pascal в части функций, необходимо к программе подключить модуль Math. То есть в начале раздела описаний надо записать строку

```
Uses Math;
```

Более того, любую функцию можно вычислить с помощью четырех арифметических операций итерационными методами или разложением в ряды.

Требования к лабораторным работам.

1. Не забывайте, что в языке Паскаль используются только латинские буквы: нет ни кириллицы (кроме строк вывода и комментариев), ни греческого алфавита.
2. Не надо дополнительно преобразовывать выражения. То есть все вычисления надо выполнять именно по тем формулам, которые приведены.
3. Числа должны быть выведены с фиксированной точкой, и количеством цифр после точки 3-6, в зависимости от размера числа или от заданной точности, с кратким пояснением выводимого числа, например:

```
c=    1.2345
Альфа  =  0.012345
```

Варианты заданий приведены в табл.2.

Таблица 2. Варианты заданий

№ вар.	Вычислить выражение	При заданных значениях
1	$f(t)=\frac{x}{t}+\frac{t}{y}+\frac{y}{x}$	$y=\sqrt{\frac{t^2+x}{\ln tx}}$; $x=\arccos t+9{,}9$; $t=0{,}11$
2	$\varphi(t)=x^3e^{-xy}\frac{\sqrt{0{,}8\,xyt}}{\cos x\cdot\ln y}$	$y=\frac{t+4}{\sin tx}$; $x=\sin t\cdot 1{,}8$; $t=0{,}392$
3	$f(y)=e^{-z}\frac{\operatorname{tg} xy+y^z}{z\cdot\cos^2 xy}$	$x=0{,}1\sqrt{zy}\cdot\frac{\sin y}{y}$; $z=0{,}3\cdot\ln y$; $y=2{,}52$
4	$\beta(t)=\frac{\omega}{v}+\sqrt{\left\lvert\frac{\sin t\omega}{\cos tv}\right\rvert}$	$\omega=\operatorname{tg}^2 t$; $v=\operatorname{ctg}^2 t+0{,}5$; $t=0{,}15$
5	$z(g)=\frac{g^2+t^2}{\sin vt}+\frac{v^2+g^2}{\cos gt}$	$v=\sqrt{g^2+t^2}$; $t=0{,}5\operatorname{tg}^2 g$; $g=9{,}8$
6	$\alpha(p)=10^{-p}\sqrt{\sin\omega p+\cos\gamma p}$	$\omega=2\gamma p^2$; $\gamma=\arcsin^3 p$; $p=0{,}15$
7	$\omega(t)=\frac{e^{-abt}\cdot\operatorname{tg}^3 bt}{1+\cos at}$	$a=\ln bt+2$; $b=\sqrt{\frac{t+1}{t}}$; $t=1{,}11$
8	$\varphi(\omega)=\arcsin\left(\omega\frac{pq}{p+q+1}\right)$	$p=\lg\omega q$; $q=1+\frac{1}{\omega^2}$; $\omega=0{,}28$
9	$\gamma(a)=a+\left(\frac{\sin abc}{a+b+c}\right)^{0{,}48}$	$b=\frac{0{,}34}{a+c}$; $c=0{,}9+a^2$; $a=1{,}98$
10	$Q(\gamma)=\lg\alpha\beta\gamma+10^{-\alpha\beta\gamma}$	$\alpha=\sqrt{\beta^2+\gamma^2}$; $\beta=0{,}8\gamma^3$; $\gamma=1{,}33$
11	$g(x)=e^{-zyx}\frac{x^2+y^2+z^2}{\sqrt{xyz}}$	$z=\ln\frac{x}{y}$; $y=0{,}55\sqrt{x}$; $x=1{,}19$
12	$\alpha(t)=\sin\frac{\omega\rho}{t}\sqrt{\frac{\omega+\rho+t}{\omega\rho t}}$	$\omega=\rho\sqrt{\frac{1}{t}}$; $\rho=10^{-t}$; $t=1{,}58$

№ вар.	Вычислить выражение	При заданных значениях
13	$\beta(x)=\dfrac{\cos^2\frac{a}{bx}+2}{a^2+\sin bx}+\dfrac{bx}{a}$	$a=\ln\sqrt{\frac{b}{x}}$; $b=2e^{-x}$; $x=2{,}35$
14	$\gamma(y)=\mathrm{tg}\frac{y}{b}+\mathrm{tg}\frac{b}{c}+\mathrm{tg}\frac{c}{y}$	$b=\dfrac{\sin c}{\cos y}$; $c=4\cdot\sqrt{\frac{0{,}8}{y}}$; $y=2{,}25$
15	$p(z)=\dfrac{\ln c+\lg bz}{b^2+c^2+z^2}$	$c=\sqrt{\dfrac{\sin z}{\cos b}}$; $b=\arcsin z$; $z=0{,}82$
16	$S(g)=\lg\dfrac{\omega^3}{dg}+\lg\sqrt{\dfrac{\omega d}{g^3}}$	$d=\omega^2+0{,}7g$; $\omega=10^{-g}\cdot g^2$; $g=1{,}83$
17	$\varphi(\omega)=\dfrac{\mathrm{tg}\,\omega x}{\sin tx+1}\cdot\dfrac{\omega}{t}$	$x=\sqrt{\ln\omega t+1}$; $t=\dfrac{1+\omega}{1-\omega}$; $\omega=0{,}87$
18	$\gamma(p)=\lg pq+\ln\sqrt{pg}$	$q=e^{-p+1}+e^{-g+1}$; $g=\mathrm{tg}\sqrt{p}$; $p=0{,}65$
19	$g(s)=\dfrac{s^2+a^2}{s^2-b^2}\cdot ab$	$a=0{,}7\sqrt{b^2+s^2}$; $b=10^{s-1}$; $s=1{,}35$
20	$t(h)=e^{-hp}+10^{-hg}$	$p=\dfrac{\sin^2 hg}{h^2+g^2}$; $g=0{,}8\arcsin\frac{1}{h}$; $h=2{,}4$
21	$u(\omega)=\dfrac{y^2-\omega^2}{\cos y\omega}+\dfrac{x^2-\omega^2}{\sin x\omega}$	$x=\dfrac{\sqrt{y+\omega}}{\mathrm{tg}\,\omega y}$; $y=\omega^2-2{,}5$; $\omega=2{,}1$
22	$z(\alpha)=\sqrt{\alpha+b+c}\cdot e^{-\alpha bc}$	$b=\ln^2(c+1)-\lg^2(\alpha+1)$; $c=\sqrt{\alpha}$; $\alpha=1{,}2$
23	$N(\beta)=\dfrac{\arcsin\beta p}{\arccos\beta g}+\sqrt{p+g}$	$p=1-e^{-\beta g}$; $g=\sqrt{\beta^2+0{,}1\beta}$; $\beta=0{,}7$

№ вар.	Вычислить выражение	При заданных значениях
24	$R(y)=\dfrac{(x+y+z)^{0,15}}{x^2+y^2+z^2}$	$x=\dfrac{\operatorname{tg} yz}{\sin y+\cos z}$; $z=0,8+\sqrt{0,3y}$; $y=0,45$
25	$\omega(g)=\dfrac{g^2p+n^2+p^2n}{\sin gp+\cos ng+\sin pn}$	$p=\sqrt{n^3g+g^3n}$; $n=0,8e^{-2g}$; $g=0,63$
26	$\rho(c)=\arcsin\sqrt{\dfrac{abc+a^2+b^2}{1+abc+a^2+b^2}}$	$a=\dfrac{\sqrt{b^3+c^3}}{b+c+1}$; $b=\operatorname{tg}\dfrac{1}{c+1}$; $c=2,4$
27	$p(t)=\dfrac{\sqrt{xyt}+2xy}{\sqrt{x+y+t}+2yt}$	$x=\dfrac{\sin(y+t)}{\cos(y-t)}$; $y=\ln 2t$; $t=1,2$
28	$\chi(m)=\ln\dfrac{m^2+p^2+c^2}{mpc+1}$	$p=\dfrac{2cm+1}{c^3+m^3+1}$; $c=m+\sqrt{m}$; $m=4,7$
29	$\beta(d)=\dfrac{e^{-kd}}{kdz+2z^3}$	$k=\dfrac{\sin d}{\cos z}$; $z=\operatorname{tg}\dfrac{d}{d+1}$; $d=0,15$
30	$\xi(l)=\ln\dfrac{l^2+fg}{l^2+f+g+1}+$ $+\lg\dfrac{f^2+gl}{g^2+l+f+1}$	$f=\sqrt{\dfrac{2g^2+3l^2}{gl+1}}$; $g=1,3\sin\dfrac{1}{l}$; $l=2,7$

Лабораторная работа № 2. Ветвящиеся алгоритмы

Логические выражения используются не только для решения задач булевой алгебры, но и для ветвления программы в логических и циклических операторах. Причем последний вариант использования логических выражений применяется наиболее часто.

Логические выражения состоят из логических констант, переменных и отношений, соединенных логическими операциями. В простейших случаях в операторах используют отношения: два выражения, соединенных знаком отношения (<, >, >=, <=, =, <>), например $I > 20$. Но иногда возникают условия, требующие использования более сложных логических выражений.

Пример. На плоскости задана фигура (рис.21): усеченный круг. Вводится точка с координатами X,Y. Определить, принадлежит введенная точка фигуре или нет. В результате выводится: «Введенная точка принадлежит фигуре» или «Введенная точка фигуре не принадлежит».

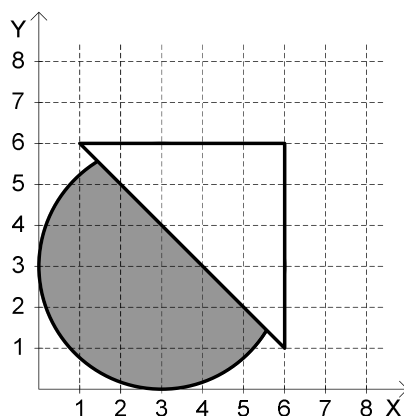


Рис. 21. Пример фигуры.

Для определения вхождения точки в круг можно использовать формулу окружности:

$$(X-X_0)^2+(Y-Y_0)^2=R^2$$

Соответственно изменив знак = на < (или <= – всё равно, так как на границах фигуры точки не проверяются) получим условие вхождения точки в круг с координатами центра (3,3) и радиусом 3:

$$(X-3)^2+(Y-3)^2<9$$

Кроме этого область, занятая треугольником, так же не входит в закрашенную область, то есть полуплоскость над прямой $Y=-X+7$ фигуре не принадлежит. Условия нахождения точки внутри круга и под прямой должны выполняться одновременно. Для этого необходимо использовать логическую операцию *AND*.

Таким образом, логическое выражение

$$(X-3)^2+(Y-3)^2<9 \quad AND \quad Y<-X+7$$

примет истинное значение, если точка входит в закрашенную область, иначе ложное. Тогда в логическом операторе по прямой ветви Then выводится «Введенная точка принадлежит фигуре», а по ветви Else «Введенная точка фигуре не принадлежит».

Но можно и поменять ветви местами, тогда при вхождении точки в фигуру логическое выражение должно принимать ложное значение. Тривиальный вариант: поставить перед предыдущим выражением знак отрицания *NOT*. Но более наглядным решением будет составление выражения с условием невхождения точки в фигуру. Здесь должно выполняться хотя бы одно из условий: точка не входит в круг или точка лежит над прямой, соответственно, логическое выражение примет вид:

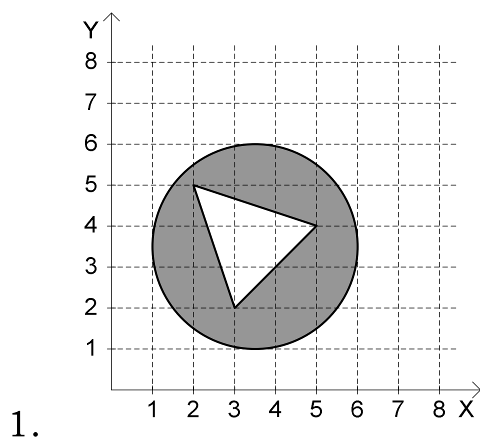
$$(X-3)^2+(Y-3)^2>9 \quad OR \quad Y>-X+7$$

При выполнении лабораторной работы составить два варианта программы (без использования операции *NOT*) для фигуры, соответствующей варианту задания.

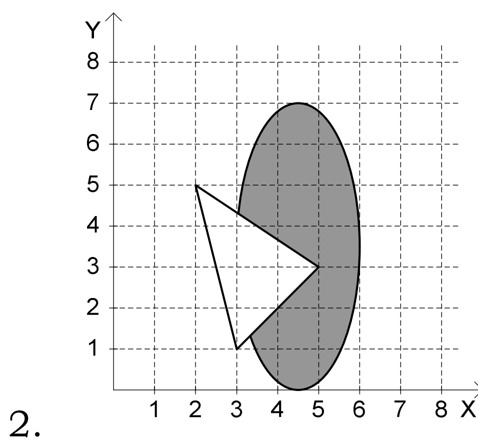
Примечание. При оформлении алгоритма длинные записи, например формулу логического выражения, можно выносить в качестве комментария.

Варианты заданий приведены в табл.3.

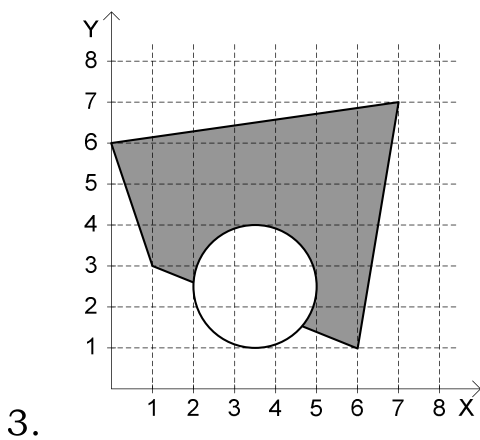
Таблица 3. Варианты заданий.



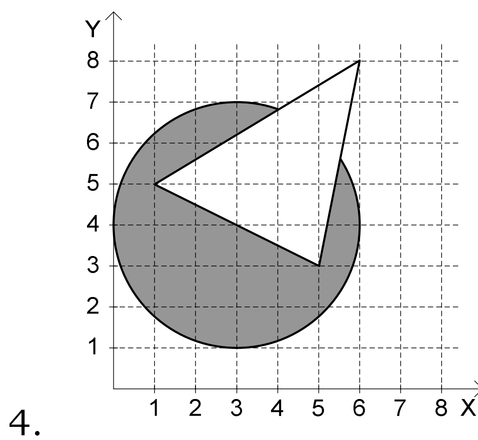
1.



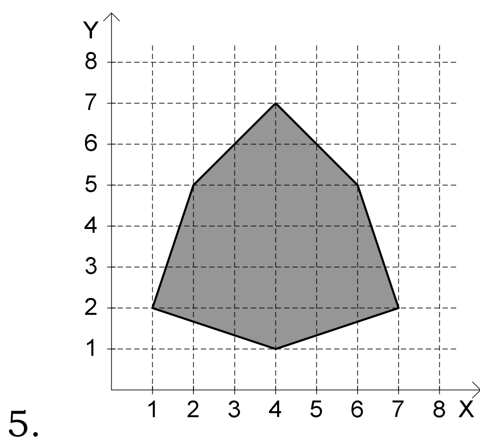
2.



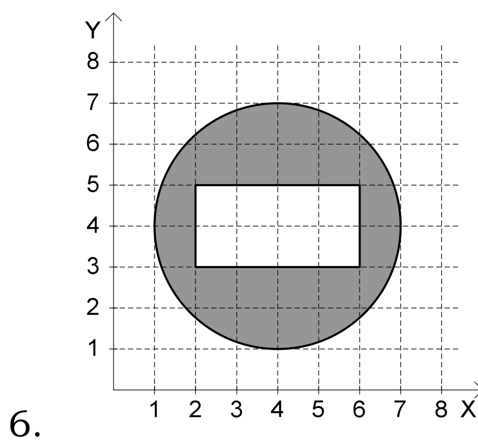
3.



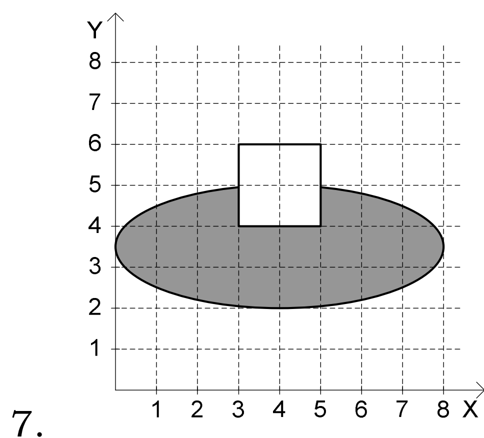
4.



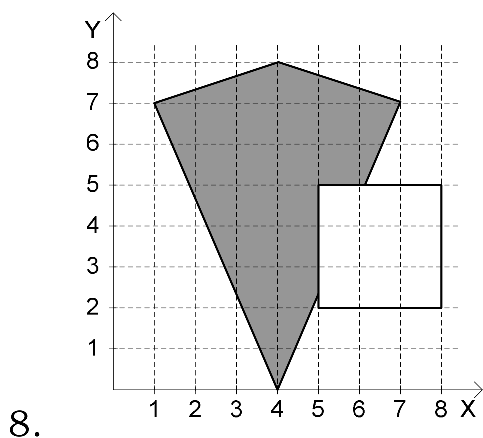
5.



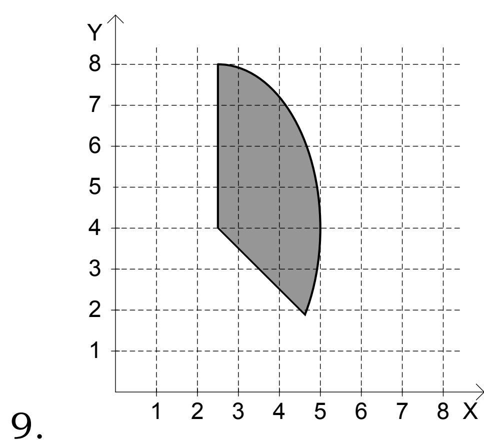
6.


Y
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 X

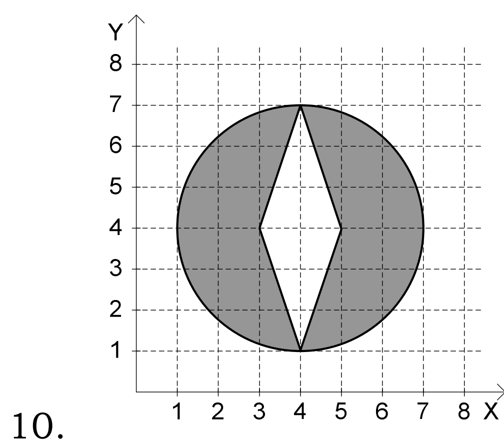
7.


Y
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 X

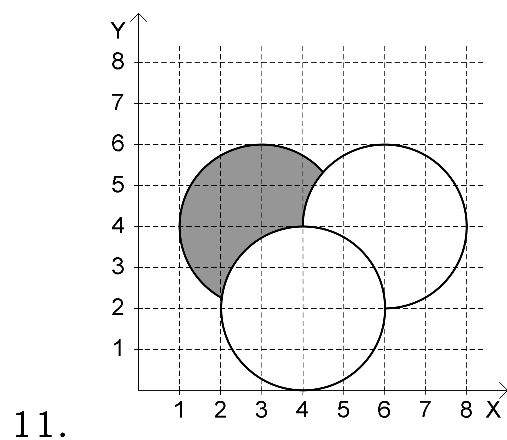
8.


Y
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 X

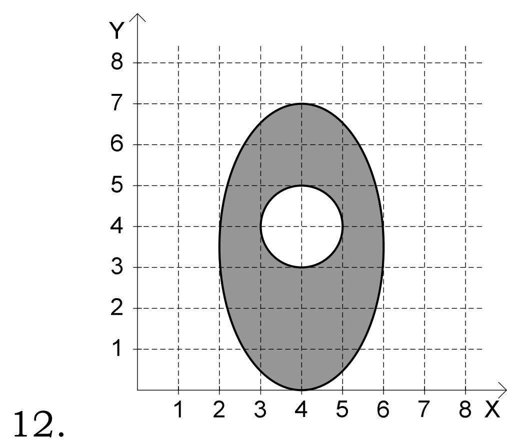
9.


Y
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 X

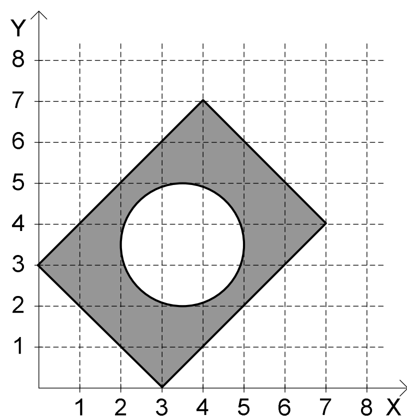
10.

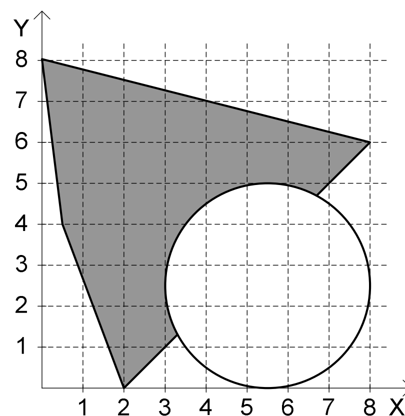

Y
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 X

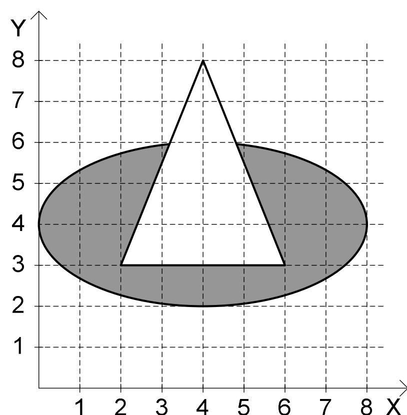
11.

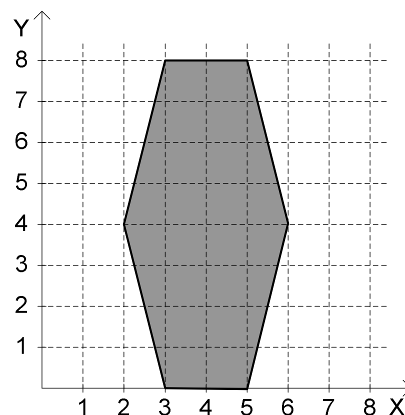

Y
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 X

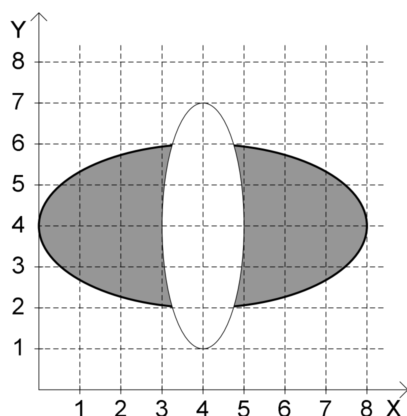
12.

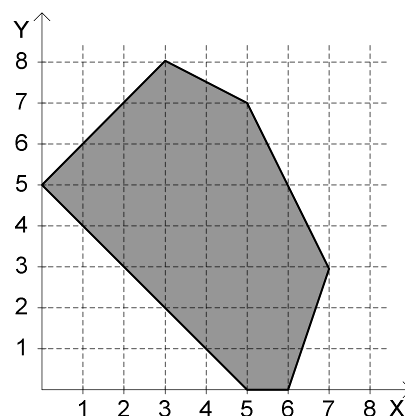
13. 

14. 

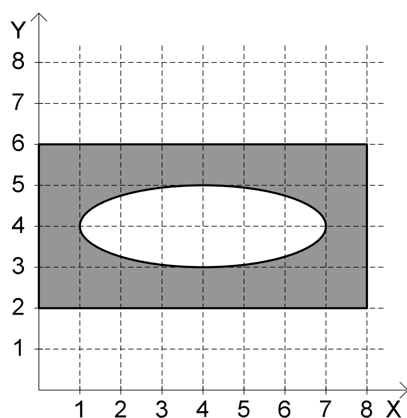
15. 

16. 

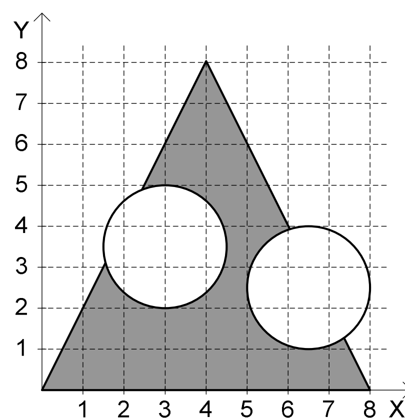
17. 

18. 

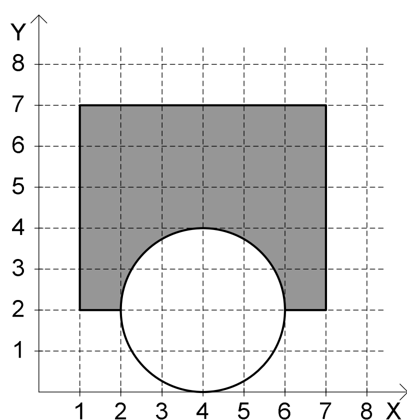
19.



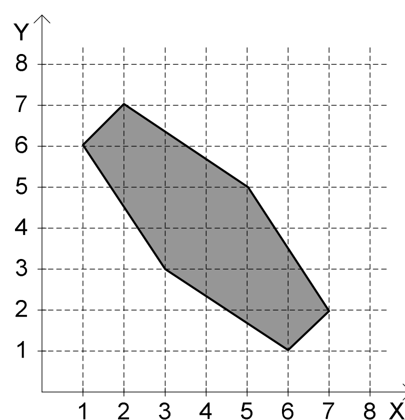
20.



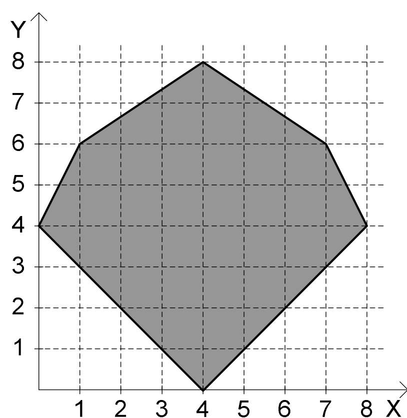
21.



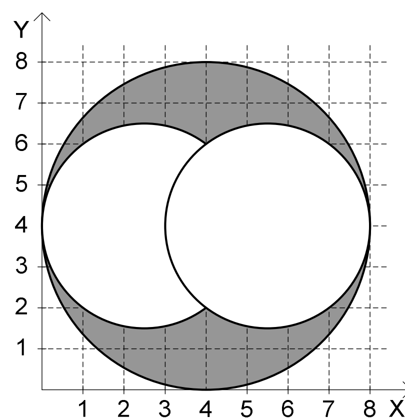
22.

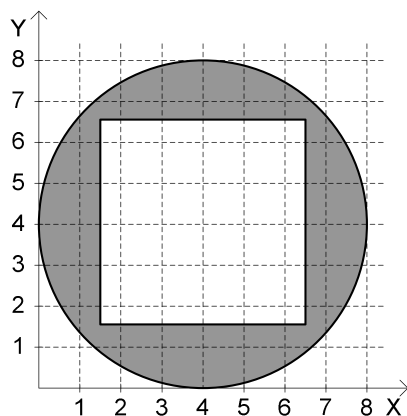


23.

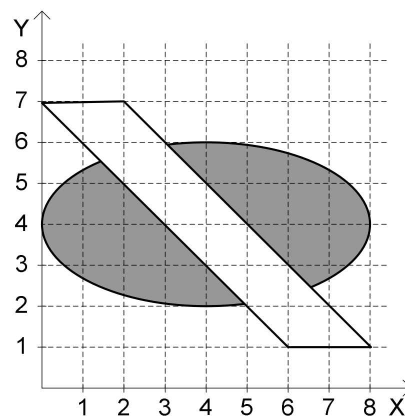


24.

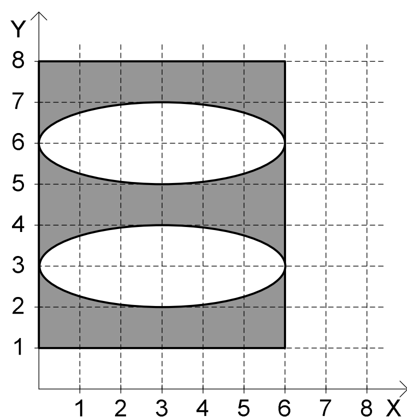




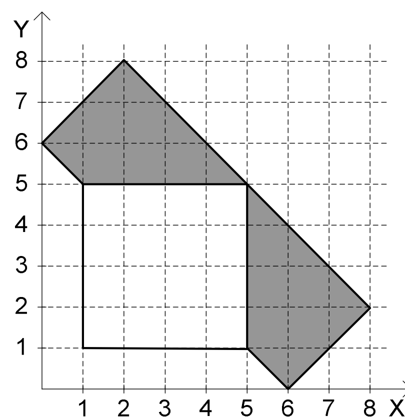
25.



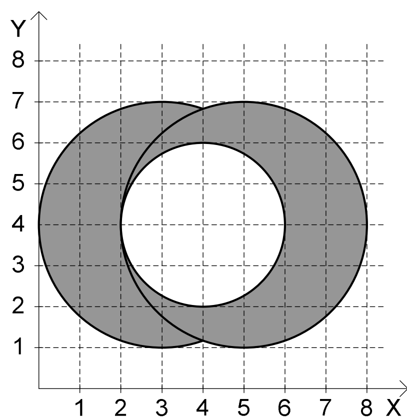
26.



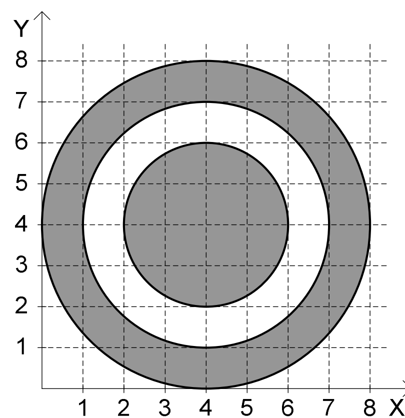
27.



28.



29.



30.

Лабораторная работа № 3. Циклы с известным числом повторений

Целью работы является освоение программирования алгоритмов с циклической структурой, когда какой-либо участок программы выполняется определенное количество раз.

Типичный пример циклического процесса – вычисление конечных сумм. При определении сумм многократно вычисляется выражение, стоящее под знаком суммы и складывается с предыдущей частичной суммой. Вычисления производятся до тех пор, пока не будут сложены выражения под знаком суммы для всех значений изменяющейся переменной.

Пример: составить программу, вычисляющую значение суммы:

$$S = \sum_{i=1}^{10} \frac{i}{i+1}$$

Прежде чем вычислять выражение под знаком суммы и очередную частичную сумму, необходимо определить начальное значение параметра цикла (в данном случае i, которое изменяется от 1 до 10 с шагом 1, то есть i будет принимать последовательно значения 1, 2, 3, 4, ..., 9, 10), и начальную частичную сумму S. Так как вычисления еще не производились, то $S = 0$.

Затем вычисляется выражение под знаком суммы для i = 1, затем i = 2, 3, ... до 10 и каждый раз складывается с предыдущей частичной суммой S_{i-1}. При этом получается новая частичная сумма S_i. После этого i увеличивается на единицу и проверяется, не стало ли $i > 10$. Если еще меньше или равно 10, то вычисляется новая частичная сумма, в противном случае вычисление суммы будет закончено, и это значение выводится на печать.

Воспользуемся стандартной схемой циклического процесса, представленной на рис.22.

Блок 1 – блок подготовки к вычислению суммы, в котором задаются начальные значения параметра цикла и частичной суммы.

В блоке 2 производится вычисление выражения, стоящего под знаком суммы и сложение с предыдущей частичной суммой S_{i-1}. В итоге получается новая частичная сумма S_i.

В блоке 3 происходит изменение параметра цикла (увеличение i на 1). Это блок подготовки к новому циклу.

Блок 4 – блок проверки окончания цикла. Необходимо проверить, стало ли i больше 10. Если стало, то цикл закончен, следующим должен выполняться блок печати. Если нет, то вычисление частичной суммы продолжается дальше, то есть выполняются блоки 2 и 3.

Проверка может осуществляться условным оператором IF, но для организации циклов в языке Паскаль специально предусмотрены три оператора цикла. Если количество повторений заранее известно, и параметр является целым числом, то целесообразно использовать оператор FOR, включающий в себя блоки 1, 3, 4. В этом случае в алгоритме можно применить блок «Модификация».

Алгоритм для примера с использованием оператора FOR приведен на рис.23.

Варианты заданий – в табл.4.

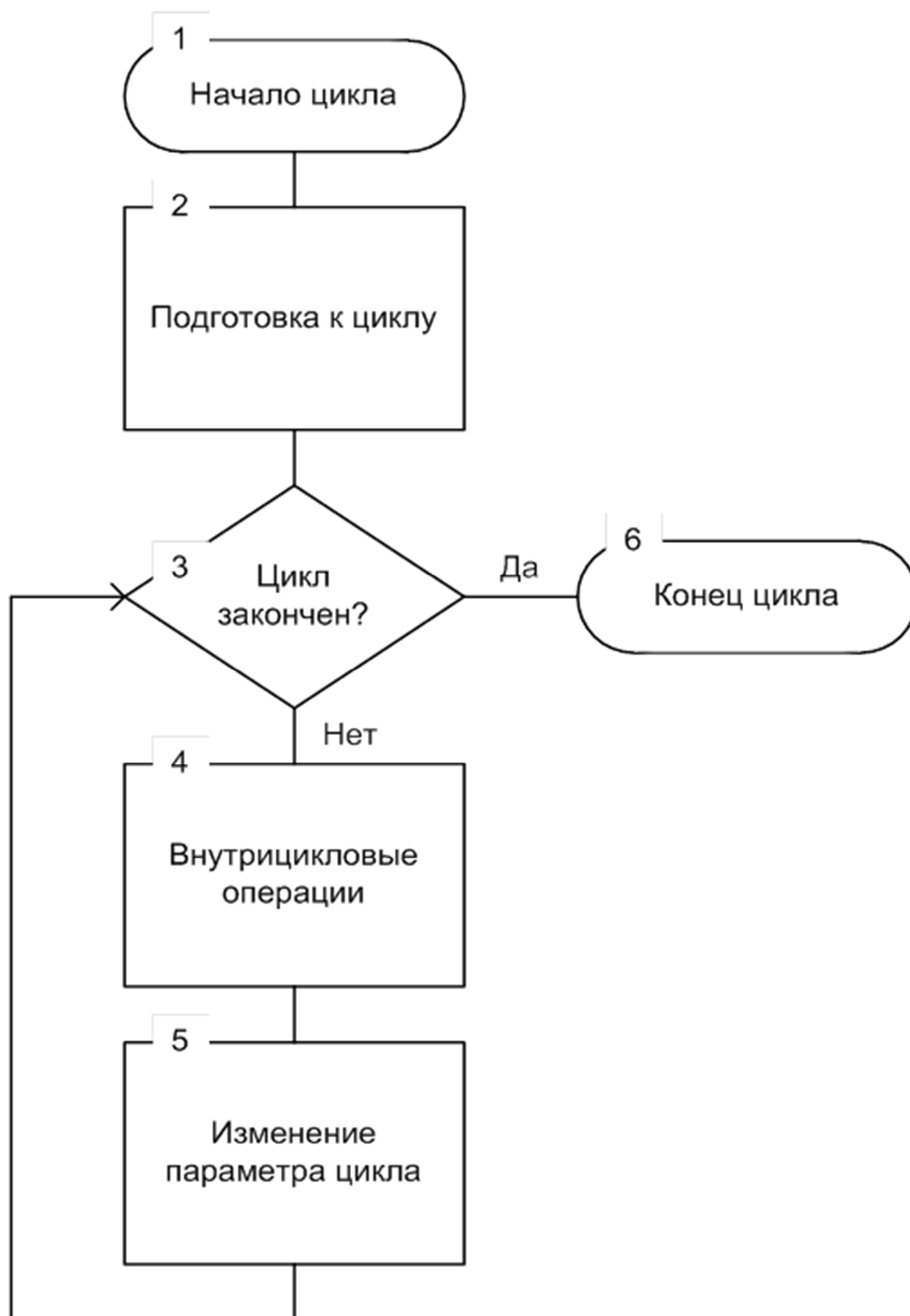


Рис. 22. Блок-схема алгоритма циклического процесса

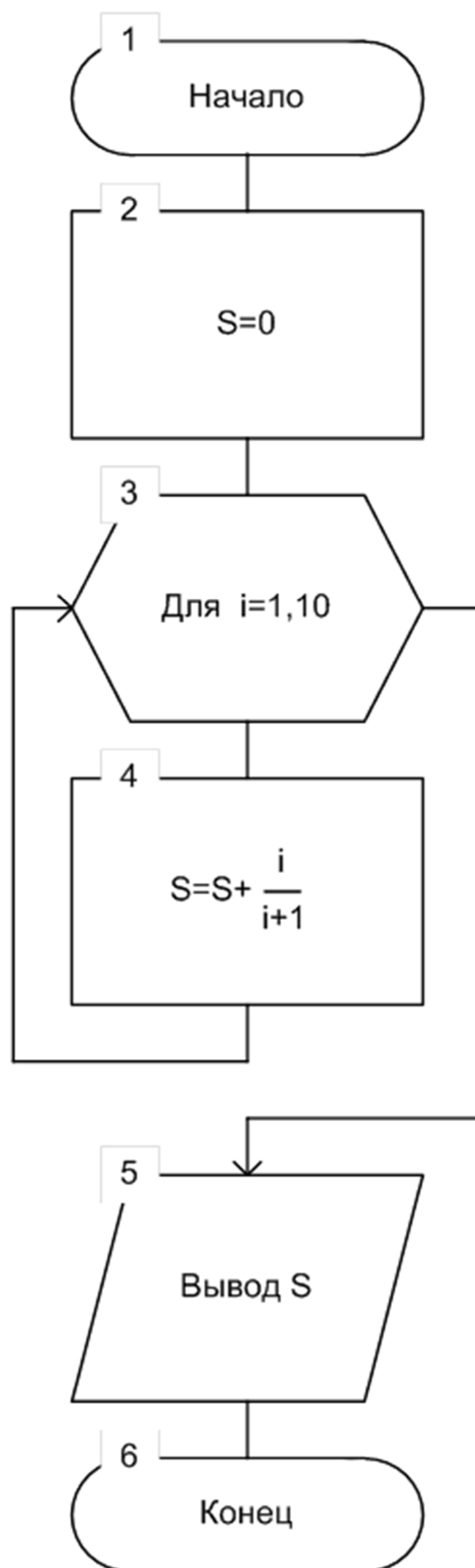


Рис. 23. Блок-схема алгоритма примера.

Таблица 4. Варианты заданий

№ вар.	Вычислить сумму	№ вар.	Вычислить сумму
1	$\sum_{i=1}^{15} \sqrt{i^2+5}$	2	$\sum_{j=1}^{20} \frac{\text{tg } j}{\text{tg}(0,5j+1)}$
3	$\sum_{k=1}^{25} \ln \frac{k}{k^2+1}$	4	$\sum_{l=1}^{30} \frac{e^{0,3l}}{\sin(l+2)}$
5	$\sum_{m=1}^{15} \text{tg}\left(e^{1/m}+1\right)$	6	$\sum_{n=1}^{20} \frac{\sin(n-2)}{\text{tg}(n+2)}$
7	$\sum_{i=1}^{25} \arccos^{0,35} \frac{i}{i+1}$	8	$\sum_{j=1}^{30} \frac{\sqrt{j^3+j+3}}{\arcsin 1/(j+1)}$
9	$\sum_{k=1}^{15} e^{0,1k}$	10	$\sum_{l=1}^{20} \frac{\sqrt{1/l}+l}{\sqrt{3l-1}}$
11	$\sum_{m=1}^{25} \ln m + e^{-m}$	12	$\sum_{n=1}^{30} \frac{e^{1/n}}{\sqrt{n^2+1}}$
13	$\sum_{i=1}^{35} 10^{0,1i} - \arccos \frac{1}{i+1}$	14	$\sum_{j=1}^{15} \frac{\lg(j+1)}{\text{tg}(j+1)}$
15	$\sum_{k=1}^{20} e^{\text{tg} 1/k}$	16	$\sum_{l=1}^{25} \frac{\arccos 0,02l}{\sqrt{2l-1}}$
17	$\sum_{m=1}^{30} \arcsin \frac{1}{m+2} \cdot \arccos \frac{1}{m+1}$	18	$\sum_{n=1}^{25} \frac{\ln n}{e^{-0,2n}}$
19	$\sum_{i=1}^{20} \cos\left(e^{0,1i}+i\right)$	20	$\sum_{j=1}^{25} \frac{\lg(j+2)}{\cos(j+1)}$
21	$\sum_{k=1}^{30} k^{0,3}+e^{0,3k}$	22	$\sum_{l=1}^{15} \frac{\text{tg}(l-3)}{\ln(l+3)}$

№ вар.	Вычислить сумму	№ вар.	Вычислить сумму
23	$\sum\limits_{m=1}^{20} \sin m \cdot \operatorname{tg} \frac{1}{m}$	24	$\sum\limits_{n=1}^{25} \frac{e^{-0{,}1n}}{\operatorname{tg}(n-1{,}5)}$
25	$\sum\limits_{i=1}^{30} \lg(i+3) \cdot e^{\sqrt{i}}$	26	$\sum\limits_{j=1}^{15} \frac{\sqrt{\lg(j+2)}}{j+2}$
27	$\sum\limits_{k=1}^{20} \sqrt{\operatorname{tg} \frac{1}{k}}$	28	$\sum\limits_{l=1}^{25} \frac{0{,}5+\lg l}{\arccos 0{,}5/l}$
29	$\sum\limits_{m=1}^{30} \cos 10^{-0{,}1m}$	30	$\sum\limits_{n=1}^{35} \frac{1}{\sqrt{\lvert\operatorname{tg}(n+1)\rvert}}$

Лабораторная работа № 4. Циклы с заранее неизвестным числом повторений

Целью работы является освоение программирования алгоритмов с циклической структурой и выхода из цикла по условию, не зависящему от количества циклов. Примером такой задачи является вычисление суммы с бесконечным верхним пределом.

Проверка цикла осуществляется следующим образом. Так как **выражение под знаком суммы постепенно убывает** с ростом слагаемых в сумме (условие сходимости), то наступает момент, когда очередное слагаемое станет меньше наперед заданного числа ε (грубо говоря, точности вычисления сумм), и остальные слагаемые будут мало влиять на конечный результат. Поэтому, когда выражение под знаком суммы $| f(i) |$ будет меньше ε, то вычисления прекращаются и предполагается, что сумма найдена с заданной точностью.

Так как количество слагаемых заранее неизвестно, то циклом FOR пользоваться нельзя. Для этих целей предназначаются циклические операторы WHILE и REPEAT. Необходимо помнить, что у этих операторов параметр цикла автоматически не изменяется, и его надо менять принудительно. Поэтому при составлении блок-схемы алгоритма блок «Модификация» не используется.

При вычислении суммы должен вычисляться факториал по формуле:

$$j! = \prod_{m=1}^{j} m$$

Где **П** – знак произведения (аналогично знаку суммы), то есть 5! = 1· 2· 3· 4· 5 = 120. Факториал можно вычислить отдельным циклом, а можно и в цикле вычисления суммы. Для этого вводится дополнительная переменная, например $f = j\,!$, и затем в цикле умножается на текущее значение j.

Кроме значения суммы на печать полезно вывести значение счетчика циклов, то есть узнать, из скольких слагаемых состоит сумма.

Примечание.

1. В языке Турбо Паскаль под переменные типа INTEGER выделяется два байта, и допустимые для них значения находятся в диапазоне только от -32768 до 32767. Поэтому число 10!, реально равное 3628800, в этом случае будет представлено как 24320. Таким образом, выражение под знаком суммы может никогда и не стать меньше заданной точности. Для работы с большими целыми числами рекомендуется использовать вещественный тип REAL с диапазоном представления от $2.9 \cdot 10^{-39}$ до $1.7 \cdot 10^{38}$, или, в крайнем случае, целый тип LongInt с диапазоном от –2.147.483.648 до 2.147.483.647.
2. Значение ε должно быть задано числом в экспоненциальной форме (в форме с плавающей точкой). Не допускается задавать его выражением.
3. Вывод суммы должен соответствовать заданной точности. Нет смысла выводить десять знаков после точки, если расчеты производятся с тремя знаками после точки. И наоборот – зачет считать с точностью 3 знака, если выводится только 1.

Варианты заданий приведены в табл.5.

Таблица 5. Варианты заданий

№ вар.	Вычислить	При x, равном	Точность вычислений ε
1	$\sum_{i=1}^{\infty} \frac{\cos ix}{i!}$	0,149	10^{-5}
2	$\sum_{j=1}^{\infty} \frac{x^2 + j^2}{j!}$	5,99	10^{-3}
3	$\sum_{i=1}^{\infty} \frac{x^i}{i! + i}$	3,1	10^{-4}
4	$\sum_{k=1}^{\infty} \frac{x^{2k}}{k! + x}$	1,91	10^{-5}
5	$\sum_{j=1}^{\infty} \frac{x^{\pi/j}}{5j!}$	1,42	10^{-3}
6	$\sum_{l=1}^{\infty} \frac{x + l}{\pi + l!}$	0,99	10^{-4}
7	$\sum_{m=1}^{\infty} \frac{\sin mx}{m!}$	1,51	10^{-5}
8	$\sum_{n=1}^{\infty} \frac{\mathrm{tg}\, x/n}{2n!}$	3,48	10^{-3}
9	$\sum_{i=1}^{\infty} \frac{e^{i/x}}{i! - 5}$	7,55	10^{-4}
10	$\sum_{j=1}^{\infty} \frac{\ln jx}{j!}$	2,15	10^{-5}
11	$\sum_{k=1}^{\infty} \frac{k + x}{k!} e^{x/k}$	0,81	10^{-3}

№ вар.	Вычислить	При x, равном	Точность вычислений ε
12	$\sum_{l=1}^{\infty} \frac{\sqrt{l^2+x^2}}{2l!}$	0,77	10^{-4}
13	$\sum_{m=1}^{\infty} \frac{x^{0,5m}}{m!}$	3,95	10^{-5}
14	$\sum_{i=1}^{\infty} \frac{\cos x/i}{i^3+i!}$	1,62	10^{-3}
15	$\sum_{j=1}^{\infty} \frac{\sqrt[3]{xj}}{j!}$	4,14	10^{-4}
16	$\sum_{k=1}^{\infty} \frac{x^{k+1}-1}{k!}$	1,24	10^{-5}
17	$\sum_{l=1}^{\infty} \frac{\lg lx}{l!-l+x}$	3,3	10^{-3}
18	$\sum_{m=1}^{\infty} \frac{\sin^2(x+m)}{m^2+m!}$	2,8	10^{-4}
19	$\sum_{i=1}^{\infty} \frac{x^3+i^2-2}{i!}$	0,95	10^{-5}
20	$\sum_{j=1}^{\infty} \frac{\ln(j+x)}{j!+2}$	4,5	10^{-3}
21	$\sum_{k=1}^{\infty} \frac{\sqrt{k^2+x^2}}{(k+2)!}$	0,85	10^{-4}
22	$\sum_{l=1}^{\infty} \frac{l^3-x^3}{l!}$	2,4	10^{-5}
23	$\sum_{m=1}^{\infty} \frac{e^{mx}}{3m!}$	1,7	10^{-3}

№ вар.	Вычислить	При X, равном	Точность вычислений ε
24	$\sum_{i=1}^{\infty} \frac{\sqrt{ix-1}}{x+i!}$	4,2	10^{-4}
25	$\sum_{j=1}^{\infty} \frac{jx}{j!}$	2,2	10^{-5}
26	$\sum_{k=1}^{\infty} \frac{1}{k^2+x^2+k!}$	3,1	10^{-3}
27	$\sum_{l=1}^{\infty} \frac{x+\lg l}{4+l!}$	1	10^{-4}
28	$\sum_{m=1}^{\infty} \frac{x^3+mx+3m}{m!}$	8,5	10^{-5}
29	$\sum_{i=1}^{\infty} \frac{20}{e^{ix}+i!}$	0,15	10^{-3}
30	$\sum_{j=1}^{\infty} \frac{j+\cos jx}{2+x+j!}$	2,9	10^{-4}

Лабораторная работа № 5. Средства вывода. Таблицы

При выводе больших объемов информации для удобства чтения ее необходимо оформлять в виде таблиц или графиков. Целью работы является изучение операторов ввода-вывода, вывод чисел в заданном виде и с определенной точностью, вывод последовательности чисел, оформленных в виде таблиц.

Таблица состоит из заголовка, в котором указано, что, в каком столбце расположено, и непосредственно таблицы набора значений выводимых переменных. При выводе заголовка таблицы используется текстовая информация. Поэтому, чтобы правильно напечаталась таблица, необходимо сделать ее макет.

Макет таблицы рисуется на бумаге в клетку, и каждая клетка принимается за одну позицию. При этом учитывается, где расположена таблица, то есть, сколько позиций надо отступить от левого края листа, каким образом проводятся вертикальные и горизонтальные линии (обычно вертикальные – набор знаков I или !, горизонтальные – знаки минус или подчеркивание). Определяется ширина таблицы, которая зависит от количества выводимых значений и точности, с какой эти значения выводятся (длина числа зависит от количества цифр в числе). После этого, символ за символом, в операторы вывода заносится с макета информация о том, как должен выглядеть заголовок таблицы.

Далее следует обычный циклический процесс с выводом в каждом цикле строки таблицы с рассчитанными значениями величин. Здесь оператор вывода наряду с текстовой информацией (вертикальная черта и пробелы), будет содержать и числовые значения.

После вывода таблицы ее необходимо подчеркнуть, то есть вывести заключительную горизонтальную линию, состоящую, например, из набора знаков минус.

Все кодовые таблицы символов имеют и символы псевдографики. Это такие символы, как вертикальная черта, прямой угол, перекрестье и т.д., например: │, └, ┘, ║, ╬, ╙. Если знать сочетания клавиш для символов псевдографики, то изображение таблиц получается лучше. Но это требуется не всегда, например, для данной лабораторной работы достаточно зна-

ков, таких, как латинское «I» большое или восклицательный знак «!», и тире «–» или символ подчеркивания «_».

Пример. Вывести таблицу значений функции $\sqrt{x}$ с точностью 7 знаков после запятой, причем x изменяется от 2 до 9 с шагом 1.

Блок-схема алгоритма представлена на рис. 26, полученный результат на рис. 24, вариант результата – на рис. 25.

```
-------------------
I  X  I  SQRT(X)  I
-------------------
I  2  I 1.4142132 I
I  3  I 1.7320509 I
I  4  I 2.0000000 I
I  5  I 2.2360678 I
I  6  I 2.4494896 I
I  7  I 2.6457510 I
I  8  I 2.8284273 I
I  9  I 3.0000000 I
-------------------
```

Рис. 24. Распечатка результата счета по программе для вывода таблиц.

X	SQRT(X)
2	1.4142132
3	1.7320509
4	2.0000000
5	2.2360678
6	2.4494896
7	2.6457510
8	2.8284273
9	3.0000000

Рис. 25. Вывод таблиц с использованием символов псевдографики.

Варианты заданий приведены в табл.6.

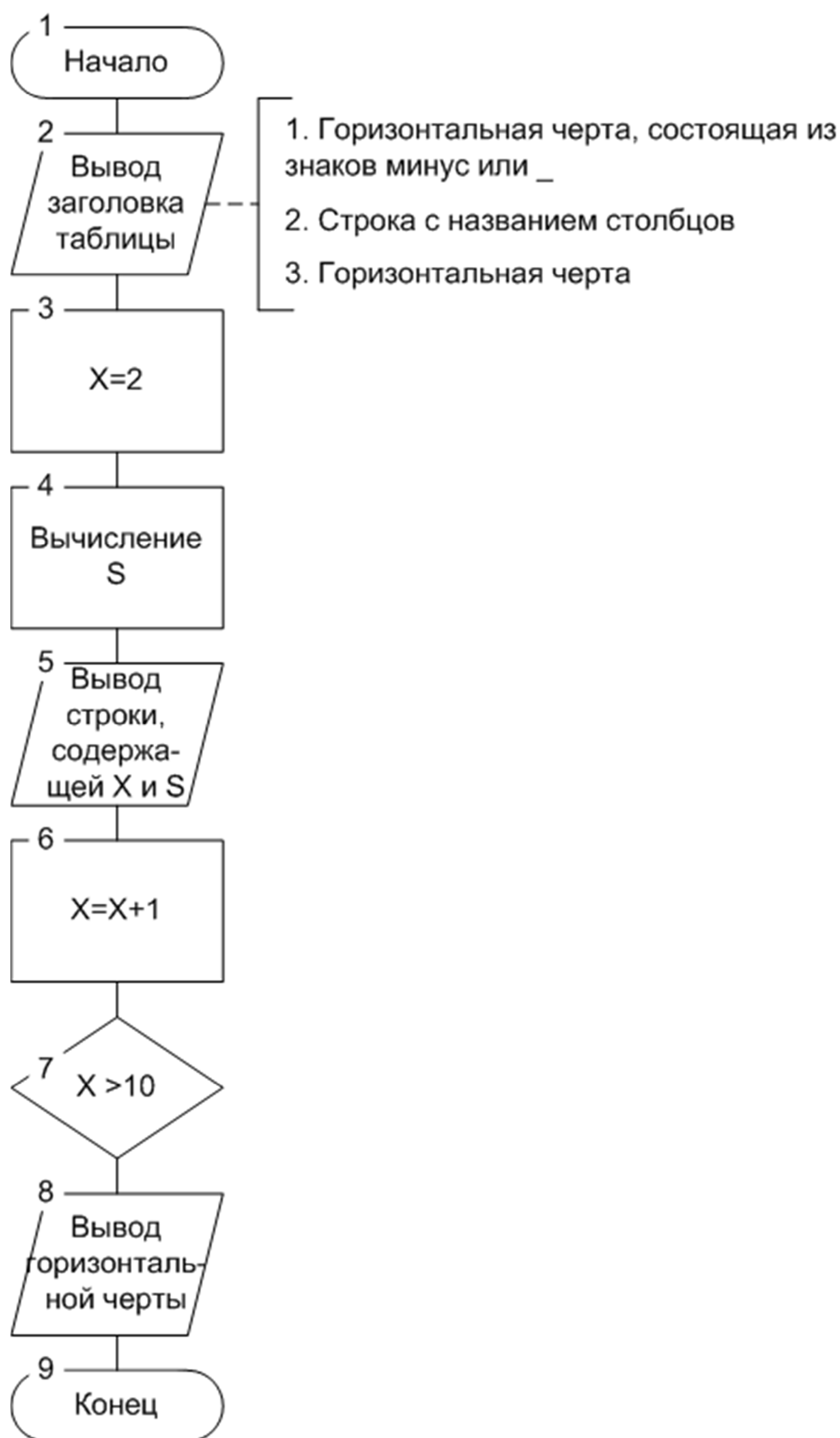


Рис. 26. Блок-схема алгоритма для примера.

Таблица 6. Варианты заданий

№ вар.	Функции	Начальное значение x	Конечное значение x	Шаг изменения x
1	$\mathrm{tg}^3 x$ и $\mathrm{ctg}^3 x$	0,2	1,7	0,1
2	$\ln x$ и $\lg x$	1	50	2
3	$\sin e^{-x}$ и $\cos e^{-x}$	0	1	0,05
4	$\sqrt[3]{x}$ и $\sqrt[5]{x}$	5	100	5
5	e^{-x} и 10^{-x}	1	2	0,05
6	$\sqrt{\ln x}$ и $\sqrt[4]{\ln x}$	1	5	0,5
7	$\cos\sqrt{x}$ и $\mathrm{tg}\sqrt{x}$	1	10	0,5
8	10^{-x} и $10^{\sqrt{x}}$	0,1	2	0,1
9	$\arcsin e^{-x}$ и $\arccos 10^{-x}$	2	5	0,2
10	$\ln\frac{1}{x}$ и $\lg\frac{1}{x}$	0,1	1	0,05
11	$\sin\frac{1}{\sqrt{x}}$ и $\cos\frac{1}{\sqrt{x}}$	0,1	2	0,1
12	$e^{x}\,\mathrm{tg}\,x$ и $e^{-x}\,\mathrm{ctg}\,x$	0,05	1	0,05
13	$\sqrt[5]{\cos x^2}$ и $\sqrt[5]{\sin x^3}$	0,05	1	0,05
14	$e^{x}\sin x$ и $e^{-x}\sin x$	0	1	0,05
15	$e^{\sqrt[3]{x}}$ и $e^{\sqrt[5]{x}}$	1	10	0,5
16	$10^{\mathrm{tg}\,x}$ и $10^{\sin x}$	0	1	0,05
17	$(x+\pi)^{0,15}$ и $(x+\pi)^{0,35}$	-1	1	0,1
18	$\ln\sqrt{x+\pi}$ и $\lg\sqrt[3]{x+\pi}$	-2	2	0,2
19	$\sqrt[15]{x+e}$ и $\sqrt[25]{x+e}$	0	100	5

№ вар.	Функции	Начальное значение x	Конечное значение x	Шаг изменения x
20	$\mathrm{sh}\sqrt{x}$ *и* $\mathrm{ch}\sqrt{x}$	1	5	0,2
21	$\dfrac{1}{\sqrt{\mathrm{sh}\, x}}$ *и* $\dfrac{1}{\sqrt{\mathrm{ch}\, x}}$	1	2	0,05
22	$\mathrm{tg}\sqrt{\sin x}$ *и* $\mathrm{ctg}\sqrt{\cos x}$	0	1	0,05
23	$x^3+\mathrm{tg}\, x$ *и* $x^2+\mathrm{ctg}\, x$	0,1	1	0,05
24	$\sqrt[3]{x^2+\sin x}$ *и* $\sqrt[3]{x^2+\cos x}$	0	2	0,1
25	$e^{-x}+x^{0,6}$ *и* $10^{-x}+x^{0,7}$	1	3	0,1
26	$\ln \mathrm{tg}\sqrt{x}$ *и* $\lg \mathrm{ctg}\sqrt{x}$	0,1	1,5	0,1
27	$\mathrm{sh}\left(x^2+1\right)$ *и* $\mathrm{ch}\left(x^3-1\right)$	0	2	0,1
28	$x^{(\pi+1)}$ *и* $x^{(e-1)}$	0,5	5	0,25
29	$\mathrm{tg}\sqrt{x^2-2}$ *и* $\mathrm{ctg}\sqrt{x^2+2}$	3	10	0,5
30	$\arcsin\dfrac{1}{x^2}$ *и* $\arccos\dfrac{1}{x^3}$	1,05	2	0,05

Лабораторная работа № 6. Двойные и кратные циклы

Целью работы является освоение программирования алгоритмов с двумя вложенными циклами. Примером такой задачи является вычисление двойной суммы.

Пример: вычислить с точностью до 0.001.

$$\sum_{i=1}^{10}\sum_{j=1}^{\infty}\frac{j^{1/i}}{j!}$$

Здесь внешней суммой является сумма по *i*, а внутренней – сумма по *j*. Можно рассматривать вычисление этих сумм отдельно, учитывая, что вычисление внутренней суммы является частью вычисления внешней суммы, то есть телом внешнего цикла. То есть цикл по *i* – внешний, по *j* – внутренний, находящийся целиком внутри текущего вычисления очередного слагаемого по *i*.

Варианты заданий приведены в табл.7.

Таблица 7. Варианты заданий

№ вар.	Вычислить	Точность вычислений ε
1	$3\sum_{i=1}^{10}\sum_{j=1}^{\infty}\frac{0{,}3ij}{i^3+j^3}$	10^{-3}
2	$\sum_{i=1}^{10} i^2 \sum_{j=1}^{\infty}\frac{i+j}{j!}$	10^{-4}
3	$0{,}3\sum_{i=1}^{20}\sum_{j=1}^{\infty}\frac{\sin\frac{ij}{\pi}}{i^5+j^5}$	10^{-3}
4	$\sum_{i=1}^{15}\cos\frac{\pi}{i}\sum_{j=1}^{\infty}\frac{\sqrt{i+j}}{(i\cdot j)^5}$	10^{-4}
5	$8\sum_{i=1}^{15}\sum_{j=1}^{\infty}\frac{\sin\frac{\pi}{ij}}{j!}$	10^{-3}
6	$\sum_{i=1}^{10} e^{-i}\sum_{j=1}^{\infty}\frac{i^{0{,}1j}}{j!}$	10^{-4}
7	$\sum_{i=1}^{15} i \sum_{j=1}^{\infty}\frac{\cos ij}{i^3+j^4}$	10^{-3}
8	$2{,}7\sum_{i=1}^{20}\ln i\sum_{j=1}^{\infty}\frac{\sqrt{i+j}}{j!}$	10^{-4}
9	$\sum_{i=1}^{12} i^2 \sum_{j=1}^{\infty}\frac{\mathrm{tg}\frac{1}{ij}}{j^3}$	10^{-3}
10	$\sum_{i=1}^{10}\cos 2i\sum_{j=1}^{\infty}\frac{i^2+0{,}5j}{j!}$	10^{-4}

№ вар.	Вычислить	Точность вычислений ε
11	$1{,}4\sum_{i=1}^{8} 10^{-i} \sum_{j=1}^{\infty} \frac{e^{-i/j}}{i^3 + j^2}$	10^{-3}
12	$\sum_{i=1}^{6} \sum_{j=1}^{\infty} \frac{j + \sin 0{,}3ij}{i - j^4 + 2}$	10^{-4}
13	$\sum_{i=1}^{8} i \sum_{j=1}^{\infty} \frac{i - 2j - 4{,}3}{(ij)^3}$	10^{-3}
14	$6{,}4\sum_{i=1}^{10} \frac{1}{i+1} \sum_{j=1}^{\infty} \frac{e^{i/j}}{i^2 + j^2}$	10^{-4}
15	$\sum_{i=1}^{5} e^{-i} \sum_{j=1}^{\infty} \frac{\sqrt{i^2 + j^2 + 1}}{j!}$	10^{-3}
16	$\sum_{i=1}^{7} \sum_{j=1}^{\infty} \frac{\sqrt{i} + 2\sqrt{j}}{i j^3}$	10^{-4}
17	$2{,}3\sum_{i=1}^{9} \sqrt{i} \sum_{j=1}^{\infty} \frac{\arcsin \frac{1}{ij+1}}{j^4 + i^2 + 1}$	10^{-3}
18	$\sum_{i=1}^{6} i^{0{,}4} \sum_{j=1}^{\infty} \frac{10^{0{,}1ij}}{j! - i^2 + 5{,}2}$	10^{-4}
19	$\sum_{i=1}^{8} i^2 \sum_{j=1}^{\infty} \frac{i + \ln ij}{\sqrt{i} + j^3}$	10^{-3}
20	$7\sum_{i=1}^{10} \sum_{j=1}^{\infty} \frac{\mathrm{tg}\frac{1}{i} + \mathrm{tg}\frac{1}{j}}{j!}$	10^{-4}

№ вар.	Вычислить	Точность вычислений ε
21	$\sum_{i=1}^{5} \frac{1}{\sin i} \sum_{j=1}^{\infty} \frac{\sqrt{i^2 + j^2}}{(i+j)^{3,8}}$	10^{-3}
22	$\sum_{i=1}^{7} \operatorname{tg} i \sum_{j=1}^{\infty} \frac{1}{j^3 - i^2 + 10,1}$	10^{-4}
23	$0,15 \sum_{i=1}^{9} (i - 0,3) \sum_{j=1}^{\infty} \frac{\sqrt{i \cdot j}}{i^3 + 4j^2 + 4i \cdot j}$	10^{-3}
24	$\sum_{i=1}^{6} \sum_{j=1}^{\infty} \frac{i^{0,1j}}{j!}$	10^{-4}
25	$\sum_{i=1}^{8} 0,1i \sum_{j=1}^{\infty} \frac{i + j^2 - 3,2}{3j^4 + 2ij + 1}$	10^{-3}
26	$3,3 \sum_{i=1}^{10} \frac{1}{i + \pi} \sum_{j=1}^{\infty} \frac{0,8\pi^2 j - i}{i^5 + j^5}$	10^{-4}
27	$\sum_{i=1}^{5} \sqrt{i} \sum_{j=1}^{\infty} \frac{\ln ij}{j! + i}$	10^{-3}
28	$\sum_{i=1}^{7} \sum_{j=1}^{\infty} \frac{\sqrt{i + j}}{i^5 + j^4 + 2ij}$	10^{-4}
29	$5 \sum_{i=1}^{9} \cos i \sum_{j=1}^{\infty} \frac{i}{j^3 + 1}$	10^{-3}
30	$\sum_{i=1}^{6} i \sum_{j=1}^{\infty} \frac{0,1\left(e^{0,55} i + j\right)}{j!}$	10^{-4}

Лабораторная работа № 7. Сортировка массивов

Массивы, – упорядоченные по индексу элементы одинакового типа, – широко используются при составлении программ. Над элементами массивов можно производить различные операции, например сортировку.

Сортировка – перестановка объектов данного множества, в частности элементов массива, в определенном порядке. Цель сортировки – облегчить последующий поиск элементов в отсортированном множестве. Она может достигаться с помощью различных алгоритмов, причем каждый из них имеет свои преимущества и недостатки.

Методы сортировки можно разделить на две различные категории: сортировка массивов (или элементов с прямым доступом) и сортировка файлов (с последовательным доступом).

Основное требование к методам сортировки массивов – экономное использование памяти, поэтому здесь не рассматриваются методы, пересылающие элементы из одного массива в другой, вспомогательный.

Рассмотрим наиболее часто используемые простые методы сортировки массивов на примерах сортировки по возрастанию.

Сортировка простыми включениями

В этом методе элементы массива условно разделяются на готовую последовательность a_1, ..., a_{i-1} и входную последовательность a_1, ..., a_n. На каждом шаге, начиная с $i = 2$ и увеличивая i на единицу, берется i-й элемент $x = a_i$ входной последовательности и вставляется в готовую последовательность после элемента, меньшего, чем x.

При поиске подходящего места чередуют сравнения и пересылки, то есть как бы «просеивают» x, сравнивая его с очередным элементом a_j и либо вставляя x, либо пересылая a_j направо и передвигаясь налево. «Просеивание» может закончиться при двух различных условиях:

1. Найден элемент a_j меньше, чем x.
2. Достигнут левый конец готовой последовательности.

Для проверки окончания «просеивания» можно воспользоваться либо двойной проверкой с объединением логической операцией, либо приемом фиктивного элемента, – «барьера». В

этом случае устанавливают «невидимый барьер» $a_0 = x$ с расширением диапазона индексов в описании a до 0..n. При этом убирается она проверка, но добавляется одно присваивание.

Вариант сортировки по возрастанию приведен на рис.27.

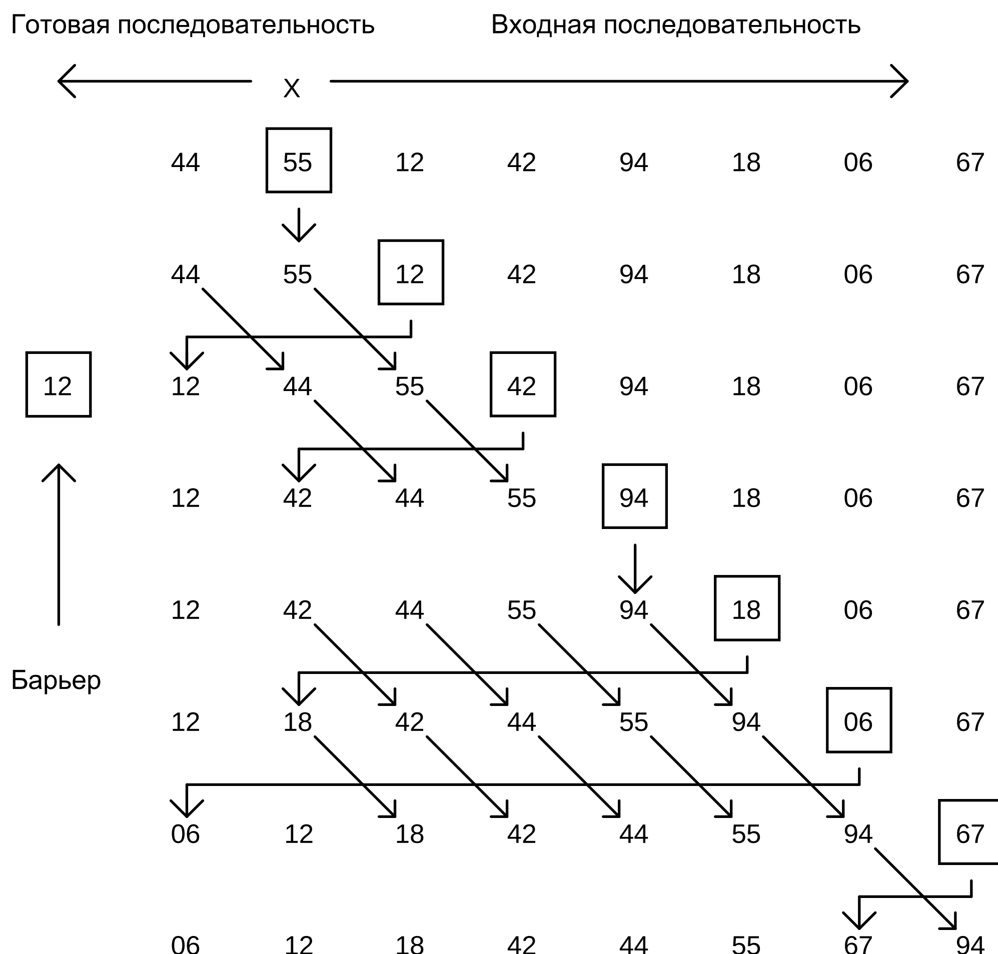


Рис. 27. Пример сортировки простыми включениями.

Сортировка бинарными включениями

Если предположить, что все перестановки n элементов готовой последовательности равновероятны, то алгоритм сортировки простыми включениями можно улучшить, ускорив поиск места включения. Для этого готовая последовательность разбивается пополам и текущий элемент сравнивается со стоящим в центре готовой последовательности элементом. Затем левый или правый интервал разбивается еще пополам, и так далее до тех пор, пока он не станет равным единице.

Хотя число сравнений и уменьшится, но количество перестановок и их порядок по сравнению с рис.8.1 не изменятся.

Сортировка простым выбором

Постоянный сдвиг большой последовательности элементов на одну позицию занимает достаточно много времени. Поэтому лучших результатов можно ожидать от метода, при котором меняются местами только два числа. На этом принципе и построен алгоритм сортировки простым выбором:

1. Выбирается наименьший элемент;
2. Он меняется местами с первым элементом;
3. Меняется на наименьший из оставшихся второй, третий и так далее, элемент.

Вариант сортировки приведен для тех же исходных данных на рис.28.

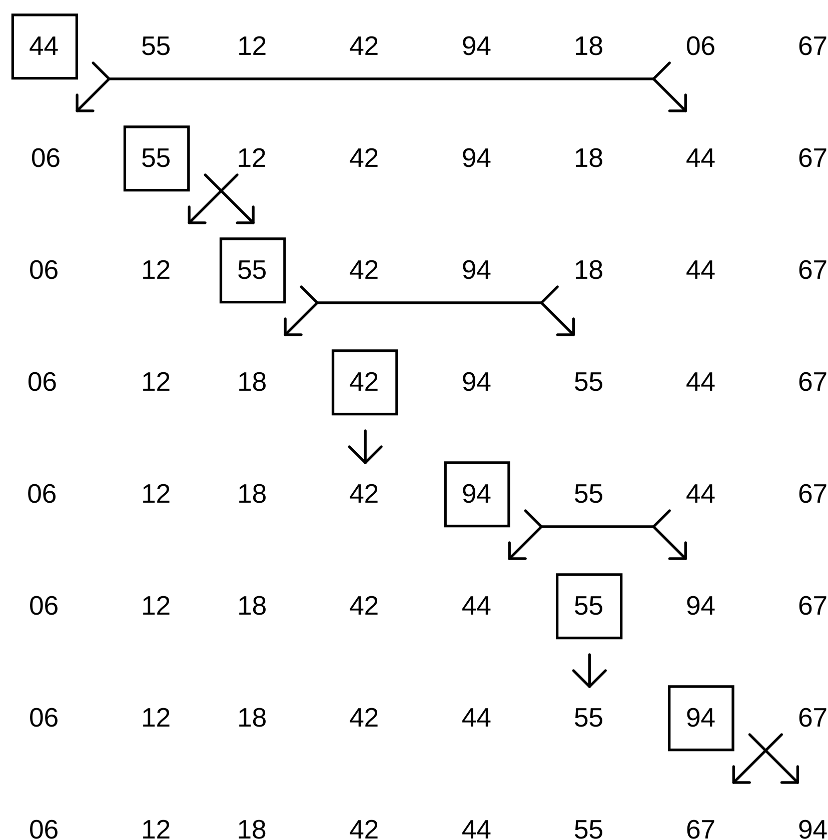


Рис. 28. Пример сортировки простым выбором.

Сортировка методом пузырька

Метод пузырька основан на сравнении и обмене двух соседних пар элементов, поэтому его иногда еще называют методом простого обмена. Если массив элементов расположить вертикально, а значение элемента сопоставить с «весом» пу-

зырька в резервуаре с водой, то каждый проход по массиву приводит к «всплыванию пузырька» на соответствующий его весу уровень, как изображено на рис.29.

Начальные значения	Номер прохода						
	2	3	4	5	6	7	8
44	06	06	06	06	06	06	06
55	44	12	12	12	12	12	12
12	55	44	18	18	18	18	18
42	12	55	44	42	42	42	42
94	42	18	55	44	44	44	44
18	94	42	42	55	55	55	55
06	18	94	67	67	67	67	67
67	67	67	94	94	94	94	94

Рис. 29. Пример сортировки методом пузырька.

Так как три последних прохода никак не влияют на порядок расположения элементов, то обычно запоминается, проводился ли на данном проходе какой-либо обмен. Если нет, то алгоритм сортировки заканчивается. Это относится и к следующему методу.

Метод шейкер - сортировки

При использовании метода пузырька возникает асимметрия, связанная с направлением сортировки. Один неправильно расположенный «пузырек» в «тяжелом» конце отсортированного массива всплывет на место за один проход, а неправильно расположенный элемент в «легком» конце будет опускаться на свое место только на один шаг на каждом проходе. Для устранения данной асимметрии и используют метод шейкер - сортировки, в котором направление проходов меняется на каждом шаге (от англ. shake – трясти) , как изображено на рис.30.

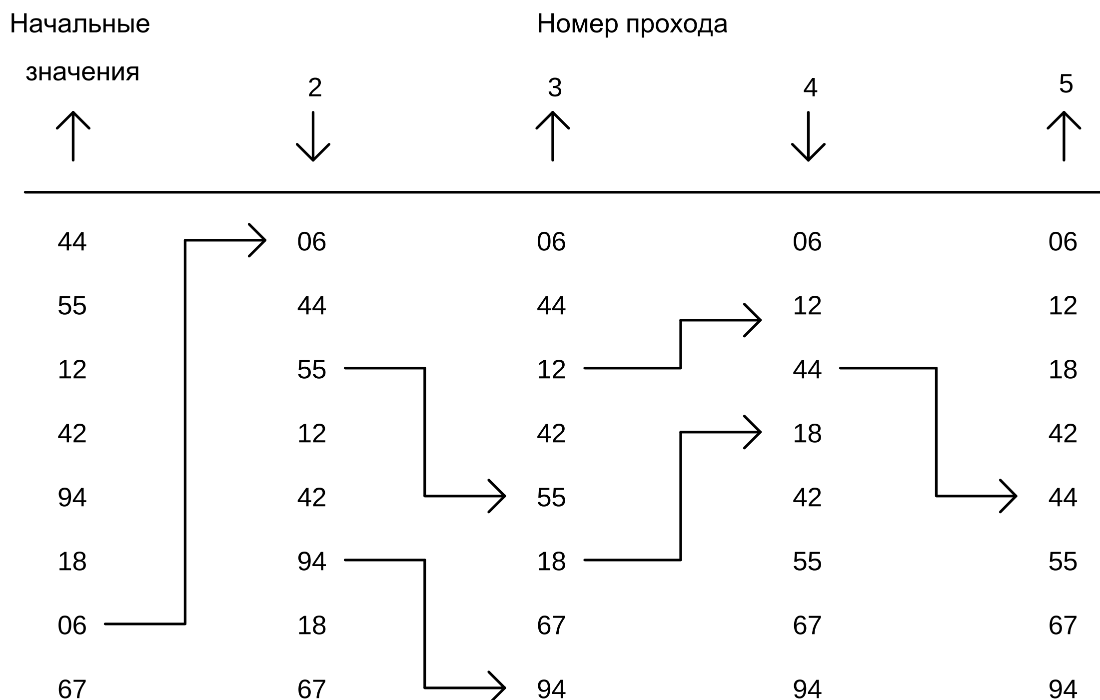


Рис. 30. Пример шейкер - сортировки.

Более сложные типы сортировок массивов элементов в данной лабораторной работе не рассматриваются.

Варианты заданий приведены в табл.8.

Составить алгоритм и программу для сортировки массивов, вводимых с клавиатуры, согласно варианту задания. В итоге выводятся исходный и отсортированный массивы.

Таблица 8. Варианты заданий

№ вар.	Метод сортировки	Максимальное кол-во элементов	Направление сортировки	Тип сортируемых элементов
1	Простые включения	12	по возрастанию	целый
2	Простые включения с барьером	10	по убыванию	вещественный
3	Простой выбор	16	по возрастанию	вещественный
4	Бинарные включения	8	по убыванию	целый
5	Метод пузырька	12	по возрастанию	литерный
6	Шейкер - сортировка	10	по убыванию	целый
7	Простые включения с барьером	14	по возрастанию	целый
8	Шейкер - сортировка	16	по возрастанию	литерный
9	Простые включения	14	по убыванию	вещественный
10	Простой выбор	14	по убыванию	литерный
11	Шейкер - сортировка	10	по убыванию	вещественный
12	Метод пузырька	14	по убыванию	целый
13	Простые включения с барьером	12	по убыванию	вещественный
14	Простые включения	16	по возрастанию	литерный
15	Простой выбор	20	по возрастанию	целый

№ вар.	Метод сортировки	Макси-мальное кол-во элементов	Направление сортировки	Тип сорти-руемых эле-ментов
16	Шейкер - сортировка	12	по убыванию	веществен-ный
17	Бинарные включения	16	по убыванию	литерный
18	Простые включения с барьером	14	по возрастанию	целый
19	Метод пузырька	10	по возрастанию	веществен-ный
20	Простой выбор	18	по убыванию	целый
21	Простые включения	10	по убыванию	веществен-ный
22	Метод пузырька	14	по убыванию	целый
23	Шейкер - сортировка	10	по возрастанию	целый
24	Простые включения с барьером	8	по убыванию	веществен-ный
25	Простые включения	14	по возрастанию	целый
26	Шейкер - сортировка	14	по убыванию	литерный
27	Метод пузырька	12	по возрастанию	веществен-ный
28	Простой выбор	14	по возрастанию	целый
29	Бинарные включения	8	по убыванию	веществен-ный
30	Простые включения с барьером	16	по возрастанию	литерный

Лабораторная работа № 8. Подпрограммы – функции

Целью данной работы является ознакомление с использованием подпрограмм – функций.

Часто некоторую последовательность действий требуется повторить в нескольких местах программы и с разными значениями. Чтобы уменьшить объем программы и время на ее набор, можно использовать структуру, присущую всем языкам программирования, – подпрограмму. Но свойство подпрограмм сокращать текст не является основополагающим. Они являются одним из фундаментальных инструментов, оказывающих влияние на стиль, качество и надежность разработки программных систем. Подпрограммы выступают как средство декомпозиции программы на логически связанные, но замкнутые компоненты, что позволяет вести ее разработку целому коллективу программистов. Так как декомпозиция существенно повышает читаемость программы, то подпрограммы, как автономные модули, используют даже тогда, когда они вызываются однократно.

В подпрограммах могут вычисляться несколько значений, например преобразование матрицы, тогда это будут подпрограммы - процедуры. Если же в подпрограмме вычисляется единственное значение, то используются подпрограммы – функции. Как следствие, работа с функциями и процедурами в языке Паскаль различна:

1. Кроме различных служебных слов в заголовке при описании функции, здесь должен быть определен тип функции, то есть возвращаемый параметр является не аргументом, а значением самой функции.
2. При описании функции должно быть хотя бы одно присваивание значения имени функции, иначе оно будет неопределенным.
3. Из основной программы обращение к подпрограмме – функции производится так же, как и к стандартным функциям, то есть указанием в выражении имени и в скобках аргументов.

Например, чтобы вычислить выражение

$$S = \frac{n! \cdot m!}{(n+m)!} ,$$

достаточно определить оператор

```
S=F(N)*F(M)/F(N+M) ,
```

при этом подпрограмму F можно записать в разделе описания функций следующим образом:

```
Function  F(K: integer): real;
   Var  Fak:real;
   Begin
     Fak:=1;
     if  K>1   then
     for  I:=2  to  K  do
          Fak:=Fak*I;
     F:=Fak
   end;
```

В лабораторной работе необходимо определить общую формулу для вычисления всех выражений. В программе вычисляются три значения *X*, *Y*, *Z* с использованием одной подпрограммы - функции. Ее аргументами будут являться как простые переменные, так и массивы:

A = { 0,12; 0,8; 0,2; 0,38; 0,11 }	*N*=5
B = { 1,5; 0,09; 0,82; 1,13 }	*N*=4
C = { 0,85; 1,4; 1,12; 3,24 }	*N*=4
D = { 0,25; 0,21; 0,12; 0,39 }	*N*=4
E = { 2,2; 3,1; 1,8 }	*N*=3

Верхние границы зависят от размера массивов *N*.

Использование массивов как параметров функции в данной работе обязательно.

Варианты заданий приведены в табл.9.

Таблица 9. Варианты заданий

№ вар.	Значение X	Значение Y	Значение Z
1	$\prod_{i=1}^{N}\sum_{j=1}^{i} B_j^2$	$\prod_{i=1}^{N}\sum_{j=1}^{i} C_j$	$\prod_{i=1}^{N}\sum_{j=1}^{i} \sqrt{D_j}$
2	$\sum_{j=1}^{N}\frac{j^2+A_j}{j!}$	$\sum_{j=1}^{N}\frac{j^3+B_j}{j!}$	$\sum_{j=1}^{N}\frac{j^4+E_j}{j!}$
3	$\max(\sqrt{D_1},...,\sqrt{D_N})$	$\max(\sqrt{D_1+X},..., \sqrt{D_N+X})$	$\max(\sqrt{D_1+Y},..., \sqrt{D_N+Y})$
4	$\min(C_1,\frac{C_2}{2},...,\frac{C_N}{N})$	$\min(C_1-X,..., C_N-X)$	$\min(C_1+Y, 2(C_2+Y), ..., N(C_N+Y))$
5	$\prod_{i=1}^{N}\sum_{j=1}^{i} B_j$	$\prod_{i=1}^{N}\sum_{j=1}^{i}(B_j-X)$	$\prod_{i=1}^{N}\sum_{j=1}^{i}(B_j-Y)$
6	$\sum_{j=1}^{N}\frac{A_j^j}{j!}$	$\sum_{j=1}^{N}\frac{(A_j-X)^j}{j!+1}$	$\sum_{j=1}^{N}\frac{(A_j-Y)^j}{j!+2}$
7	$\max\left(\frac{1}{B_1},...,\frac{N}{B_N}\right)$	$\max(C_1,...,C_N\cdot N)$	$\max(D_1^3,...,D_N^3\cdot N)$
8	$\min\left(\frac{1}{A_1},...,\frac{N}{A_N}\right)$	$\min(\frac{1}{D_1+1},..., \frac{N^2}{D_N+N})$	$\min(\frac{1}{E_1+2},..., \frac{N^3}{E_N+2N})$

№ вар.	Значение X	Значение Y	Значение Z
9	$\prod_{i=1}^{N}\frac{10+B_i}{i!}$	$\prod_{i=1}^{N}\frac{20+C_i}{i!}$	$\prod_{i=1}^{N}\frac{30+D_i}{i!}$
10	$\sum_{i=1}^{N}\prod_{j=1}^{i}\frac{i}{A_j}$	$\sum_{i=1}^{N}\prod_{j=1}^{i}\frac{i}{C_j+1}$	$\sum_{i=1}^{N}\prod_{j=1}^{i}\frac{i}{E_j+2}$
11	$\max\left(E_1,\ldots,E_N\right)$	$\max\left(\frac{E_1^2}{X},\ldots,\frac{E_N^2}{X^N}\right)$	$\max\left(\frac{E_1^3}{Y},\ldots,\frac{E_N^3}{Y^N}\right)$
12	$\min\left(D_1^2,\ldots,D_N^2\right)$	$\min\left[\left(D_1-\sqrt{X}\right)^2,\ldots,\left(D_N-\sqrt{X}\right)^2\right]$	$\min\left[\left(D_1-\sqrt{Y}\right)^3,\ldots,\left(D_N-\sqrt{Y}\right)^3\right]$
13	$\prod_{i=1}^{N}\frac{\lvert C_i\rvert}{i!}$	$\prod_{i=1}^{N}\frac{\left(C_i-X\right)^2}{i!}$	$\prod_{i=1}^{N}\frac{\lvert C_i-Y\rvert^3}{i!}$
14	$\sum_{i=1}^{N}\prod_{j=1}^{i}A_j$	$\sum_{i=1}^{N}\prod_{j=1}^{i}\left(A_j-X\right)$	$\sum_{i=1}^{N}\prod_{j=1}^{i}\left(A_j-Y\right)$
15	$\max\left(B_1^2,\ldots,B_N^2\right)$	$\max\left(C_1,\ldots,C_N\right)$	$\max\left(\sqrt{D_1},\ldots,\sqrt{D_N}\right)$
16	$\min\left(A_1,\ldots,\frac{A_N}{N}\right)$	$\min\left(\frac{C_1}{2},\ldots,\frac{C_N}{N+1}\right)$	$\min\left(\frac{E_1}{3},\ldots,\frac{E_N}{N+2}\right)$
17	$\prod_{i=1}^{N}B_i^2$	$\prod_{i=1}^{N}\left(C_i+X\right)^2$	$\prod_{i=1}^{N}\left(D_i+Y\right)^2$
18	$\sum_{i=1}^{N}A_i^3$	$\sum_{i=1}^{N}D_i$	$\sum_{i=1}^{N}\frac{1}{E_i}$

№ вар.	Значение X	Значение Y	Значение Z
19	$\max\left(E_1,\ldots,E_N\right)$	$\max\left(\frac{1}{E_1+X},\ldots,\frac{1}{E_N+X}\right)$	$\max\left(\frac{1}{\left(E_1+Y\right)^2},\ldots,\frac{1}{\left(E_N+Y\right)^2}\right)$
20	$\min\left(D_1^2,\ldots,D_N^2\right)$	$\min\left[\left(D_1-X\right)^2,\ldots,\left(\frac{D_N-X}{N}\right)^2\right]$	$\min\left[\left(D_1-Y\right)^2,\ldots,\left(\frac{D_N-Y}{N^2}\right)^2\right]$
21	$\prod_{i=1}^{N}\frac{C_i}{i}$	$\prod_{i=1}^{N}\frac{C_i-X}{i^2}$	$\prod_{i=1}^{N}\frac{C_i-Y}{i^3}$
22	$\sum_{i=1}^{N}\frac{B_i}{i}$	$\sum_{i=1}^{N}\frac{\left(B_i-X\right)^2}{i}$	$\sum_{i=1}^{N}\frac{\left(B_i-Y\right)^3}{i}$
23	$\max\left(A_1,\ldots,\frac{A_N}{N}\right)$	$\max\left(A_1,\ldots,\frac{B_N}{N^2}\right)$	$\max\left(A_1,\ldots,\frac{E_N}{N^3}\right)$
24	$\min\left(B_1^2,\ldots,B_N^2\right)$	$\min\left(C_1^3,\ldots,C_N^3\right)$	$\min\left(D_1^4,\ldots,D_N^4\right)$
25	$\prod_{i=1}^{N}\frac{A_i}{i}$	$\prod_{i=1}^{N}\frac{B_i}{i}$	$\prod_{i=1}^{N}\frac{E_i}{i}$
26	$\sum_{i=1}^{N}B_i$	$\sum_{i=1}^{N}C_i^2$	$\sum_{i=1}^{N}D_i^3$

№ вар.	Значение X	Значение Y	Значение Z
27	$\max\left(C_1^2,\ldots,\frac{C_N^2}{N}\right)$	$\max\left(C_1 - X,\ldots,\frac{C_N - X}{N}\right)$	$\max\left(C_1 + Y,\ldots,\frac{C_N + Y}{N}\right)$
28	$\min\left(B_1,\ldots,B_N\right)$	$\min\left[(B_1 - X)^2,\ldots,(B_N - X)^2\right]$	$\min\left[(B_1 - Y)^3,\ldots,(B_N - Y)^3\right]$
29	$\sum_{i=1}^{N}(D_i + 1)$	$\sum_{i=1}^{N}(D_i - X)^2$	$\sum_{i=1}^{N}(D_i + Y)^3$
30	$\sum_{i=1}^{N}\frac{(A_i + N - 5)}{5 + i}$	$\sum_{i=1}^{N}\frac{(A_i + N - X)}{X + i}$	$\sum_{i=1}^{N}\frac{(A_i + N + X - Y)}{Y - X + i}$

Лабораторная работа № 9. Подпрограммы – процедуры

В тех случаях, когда подпрограмма не возвращает никакого конкретного значения, воспользоваться подпрограммой – функцией нельзя. Обычно не используют ее и когда несколько результатов, например, при преобразованиях массивов.

При обращении к подпрограмме – процедуре просто указывается ее имя со списком фактических параметров. Примером стандартных процедур могут быть подпрограммы ввода - вывода writeln, readln и другие.

Написать программу с использованием подпрограммы-процедуры, вычисляющую три вектора или матрицы *X*, *Y*, *Z*, являющиеся комбинацией трех векторов по два:

$$X = f\left(\bar{M}_1, \bar{M}_2\right)$$

$$Y = f\left(\bar{M}_1, \bar{M}_3\right)$$

$$Z = f\left(\bar{M}_2, \bar{M}_3\right),$$

если заданы исходные вектора:

A = {3; 0; -1; 5; 7}
B = {8; 4.2; 8.8; 5.5}
C = {-1; 6; -1.8; 6.7}
D = {0.7; -1.1; 5.1; 6}
E = {-0.09; 10; 2.2; 4.5}
F = {5.5; 3.1; 2.4; 7} .

В вариантах приводится одна формула для вычисления трех массивов *X*, *Y*, *Z*:

$$F\left(P,Q\right) = f\left(\bar{M}_k, \bar{M}_l\right),$$

где соответственно k = 1, 1, 2 и l = 2, 3, 3.

Например, формула $F\left(P,Q\right) = P_i + Q_j$ с используемыми векторами *A*, *B*, *C*, приводит к трем матрицам (разные индексы i и j):

$$X_{ij} = A_i + B_j$$
$$Y_{ij} = A_i + C_j$$
$$Z_{ij} = B_i + C_j$$

Во всех вариантах i и j изменяются от 1 до 4.
Варианты заданий приведены в табл.10.

Таблица 10. Варианты заданий

№ вар.	$F\,(P,Q)$	Используемые вектора
1	$\sqrt{\dfrac{P_i+Q_j+5}{2}}$	*A, B, C*
2	$\sin\left(\dfrac{P_i}{Q_i}+\dfrac{Q_i+1}{P_i-1}\right)$	*D, E, F*
3	$\dfrac{P_i+Q_j+1}{Q_i+P_j-1}$	*A, C, E*
4	$P_i Q_i \dfrac{\sin P_i}{\cos Q_i}$	*B, D, F*
5	$\arcsin\dfrac{1}{P_i+10}+\arccos\dfrac{1}{Q_j+10}$	*B, C, D*
6	$e^{-P_i Q_i}$	*C, D, E*
7	$0{,}8\sqrt{P_i^2+Q_j^2}+1{,}8\sqrt{\dfrac{1}{P_i^2+Q_j^2}}$	*A, B, C*
8	$\sin\dfrac{P_i^2+1}{Q_i^2+1}$	*D, E, F*
9	$\dfrac{P_i^2+Q_j^2+2P_jQ_i}{4P_iQ_j+1}$	*B, C, D*
10	$\dfrac{P_i}{Q_i}+4\dfrac{Q_i+1}{P_i+1}$	*C, D, E*

№ вар.	$F(P,Q)$	Используемые вектора
11	$\dfrac{j+i}{\ln\left(P_i+2\right)+\ln\left(Q_j+2\right)}$	*A, C, E*
12	$\dfrac{i}{\arccos\dfrac{1}{P_i+Q_i}}$	*B, D, F*
13	$\sqrt{\dfrac{P_i+j}{Q_j+i}+5}$	*A, B, C*
14	$\dfrac{1}{P_i^2+Q_i^2+0{,}5}$	*D, E, F*
15	$\dfrac{\sin\left(P_i-1\right)}{\cos\left(Q_j+1\right)}$	*B, C, D*
16	$1{,}7e^{-P_i}+0{,}4e^{-Q_i}$	*C, D, E*
17	$\sqrt{P_i^2+Q_i^2+ij}$	*A, C, E*
18	$\ln\dfrac{Q_i^2+1}{P_i^2+1}$	*B, D, F*
19	$\operatorname{tg}\dfrac{Q_i+i}{P_j+j}$	*A, B, C*
20	$\ln\left(\sqrt{P_i+j+1}+\sqrt{Q_j+i+1}\right)$	*D, E, F*

№ вар.	$F(P,Q)$	Используемые вектора
21	$\dfrac{1}{P_i^2 + Q_j^3 + i + j}$	*B, C, D*
22	$3P_i^2 - 2Q_i^2 P_i - 1$	*C, D, E*
23	$i \sin P_j + j \cos Q_i$	*A, C, E*
24	$3{,}3\sqrt{P_i + Q_i}$	*B, D, F*
25	$\dfrac{\sin P_i Q_j}{P_j^2 + Q_i^2}$	*F, B, C*
26	$\dfrac{P_i}{i+1} + \dfrac{Q_i}{i-1}$	*D, E, F*
27	$\cos \dfrac{P_i + Q_j}{P_i Q_j}$	*B, C, D*
28	$\ln \dfrac{i}{(P_i + Q_i)^2}$	*C, D, E*
29	$\mathrm{tg}\, j P_i + \mathrm{tg}\, i Q_j$	*A, C, E*
30	$e^{0{,}3\sqrt{P_i^2 + Q_i^2} - i}$	*B, D, F*

Лабораторная работа №10. Строки. Перевод чисел из одной системы счисления в другую

Целью работы является освоение программирования алгоритмов с циклической структурой и выхода из цикла по условию, не зависящему от количества циклов.

Формула для перевода из одной системы счисления в другую представлена ниже:

$$x = a_{r-1} \cdot p^{r-1} + a_{r-2} \cdot p^{r-2} + \dots + a_1 \cdot p^1 + a_0 \cdot p^0 = \sum_{r=0}^{n-1} a_r \cdot p^r$$

Здесь x – получившееся число с основанием p.

a_i – цифры исходного числа.

Так как преобразования выполняются в той системе счисления, в которую осуществляется перевод, то для перевода в систему, отличную от десятичной, используется полином Горнера:

$$x = \left(\left(\dots\left(a_{r-1} \cdot p + a_{r-2}\right) \cdot p + a_{r-3}\right)\dots\right) \cdot p + a_0.$$

Для обозначения цифр, больших девяти, – в системах счисления одиннадцатиричной и более, – используются буквы латинского алфавита. Единственной «родной» для компьютера системой счисления является шестнадцатеричная. Да и то, что количество разрядов в одном байте ровно два. То есть запись чисел сокращается ровно в 4 раза. Но бывают самые разнообразные системы счисления.

Для их указания используются подстрочные числа, например

$00110101_2=53_{10}$

$ABCD_{16}=43981_{10}=1010101111001101_2$.

Но в компьютерах в текстах иногда отсутствуют подстрочные и надстрочные символы, поэтому для обозначения систем счисления используются либо буквы, либо специальные символы. Десятичная система счисления обозначается по умолчанию, то есть никак не обозначается. В Паскале для обо-

значения шестнадцатеричной системы счисления перед числом ставится знак доллара – $. В языке Си используется префикс 0x. В некоторых версиях Бейсика используется префикс «&h». В языках Ассемблера, – в них присутствуют все используемые на компьютерах системы счисления: двоичная, восьмеричная, десятичная и шестнадцатеричная, – часто в конце числа ставится буква – признак системы счисления. B (binary) – двоичная, O (octal) – восьмеричная, не ставится или D (decimal) – десятичная, H (hexdecimal) – шестнадцатеричная, но системы счисления могут обозначаться и по другому.

При вводе-выводе и использовании букв при записи недесятичных чисел лучше использовать не числовые, а строковые данные, которые затем в цикле посимвольно, согласно (13.1), переводятся в десятичное число.

В качестве максимальной границы используется функция **Length (<строка>)**, а внутри цикла каждый символ (цифра a_i) выбирается оператором выбора **CASE**.

Примечания.

1. Число в заданной системе счисления должно вводиться с клавиатуры.
2. Так как переводятся целые числа, то лучше всего использовать для результата четырехбайтный целочисленный тип данных Longint, так как под тип Integer отводится только 2 байта.
3. Длина числа должна быть 5-4 цифры (символа), причем состоять минимум из двух букв (остальные – десятичные арабские цифры).
4. При вводе недопустимых символов, например *x* или точки, должно выводиться какое-либо сообщение без вывода результата. Это легко достигается с помощью ветви **else** оператора **CASE**.

Варианты задания приведены в табл. 11.

Таблица 11. Варианты заданий.

Перевести число из системы счисления с основанием *n* в десятичную систему счисления.

№ вар.	n	№ вар.	n	№ вар.	n
1	17	11	27	21	22
2	18	12	28	22	23
3	19	13	29	23	24
4	20	14	30	24	25
5	21	15	31	25	26
6	22	16	17	26	27
7	23	17	18	27	28
8	24	18	19	28	29
9	25	19	20	29	30
10	26	20	21	30	31

Лабораторная работа № 11. Работа с файлами и строками

Целью работы является приобретение навыков работы с файловыми структурами при работе с модулем System (для Pascal ABC – PABCSystem). Модуль Dos в данной лабораторной работе не рассматривается. А так же изучаются операции работы со строками.

Прежде, чем использовать файлы, им надо поставить в соответствие файловые переменные процедурой Assign. Например:

```
Assign (f,’D:/student/002175/myfile.dat’);
Assign (f,’myfile.dat’);
```

Во втором случае файл находится в текущем каталоге. Это единственная структура, которая подчиняется непосредственно операционной системе, всё остальное (кроме комментариев) в тексте программы принадлежит правилам языка Паскаль.

Эта процедура должна стоять первой при начале работы с файлом. Сама же файловая переменная описывается в разделе описания переменных строкой вида

```
Var <список_файловых_переменных>: FILE OF <тип_компонент>
```

Так как строки являются особенными структурами данных, то для их хранения используется специальное описание файловых переменных:

```
Var  < имя_файловой_переменной >: text;
```

Например:

```
Var  File, OutF: text;
```

По большому счету все файлы можно либо создавать, либо читать. В первом случае используется процедура открытия файла для записи Rewrite, во втором – открытие файла для чтения Reset.

В дальнейшем в первом случае в них можно записывать обычной процедурой Write, во втором – читать процедурой

Read. Если же используются текстовые файлы, то так же можно использовать процедуры WriteLn и ReadLn.

Для процедур Write/ WriteLn и Read/ ReadLn консоль является стандартным устройством ввода-вывода, поэтому имена файлов для клавиатуры и экрана монитора можно не указывать. Во всех остальных случаях в строке ввода-вывода на первом месте должна стоять файловая переменная. Например:

```
WriteLn(OutF,'Я помню чудное мгновенье');
```

Над символьными массивами разрешены только операции сравнения. Над строками разрешено гораздо больше операций. Они могут быть оформлены не только в виде знаков, таких как + (конкатенация), но и в форме процедур или функций.

Например, функция Copy (<строка>, <номер первого символа>, <количество символов>) позволяет копировать или выделять фрагмент строки.

Функция Pos (<искомая подстрока>, <строка>) позволяет произвести поиск определенного фрагмента в некоторой строке и определить номер символа, с которого начинается вхождение подстроки.

Функция Length (<строка>) позволяет определить не предельную, а фактическую длину строки. Результат – целое число.

Процедура Delete (<строка>, <номер первого удаляемого символа>, <количество символов>) удаляет в исходной строке фрагмент определенной длины.

Прочие функции (но не процедуры) приведены в Приложении А.

Более того, можно получить доступ к любому символу строки, указав ее в виде элемента массива с индексом. В описании строки можно указывать предельное значение количества символов, но можно и не указывать:

```
Var st,at: string;
   St1,st2:string[20];
```

Если длина не задана, то по умолчанию (автоматически) принимается максимально возможная – 255 символов.

Задание

В данной лабораторной работе используются две программы: одна для формирования текстового файла с расширением .txt, другая для работы с ним.

Сформировать файл с заданной структурой, состоящий из 8-16 строк, каждая из которых состоит из нескольких слов. Для этих целей лучше всего использовать фрагмент стихотворения. Во второй программе требуется прочитать данные из файла, обработать их определенным образом и записать в тот же файл.

Варианты задания приведены в табл. 12.

Таблица 12. Варианты заданий

№ Вар.	Задание
1.	Вывести в файл количество слов, начинающихся с разных букв, упорядоченных по алфавиту. То есть результат должен содержать таблицу, в первой графе которой название буквы, во второй – количество слов, начинающихся с этой буквы. Если слов, начинающихся с данной буквы, нет, то строка пропускается.
2.	Использовать шифрование методом простой перестановки с длинной блока 3. Размер текста должен быть кратен 3, в противном случае текст дополняется пробелами. В данном случае местами меняются только крайние буквы, средняя остается на месте. Пример: До шифрования: |МАМ|А М|ЫЛА| РА|МУ | После шифрования: |МАМ|М А|АЛЫ|АР | УМ|
3.	Подсчитать количество слов, пробелов и знаков препинания в файле. Записать результат в файл в виде «количество слов= <число>, количество пробелов= <число>, количество знаков препинания= <число>».
4.	Провести анализ вхождения букв начала алфавита А, Б, В, Г. То есть определить отношение появления этих букв ко всем буквам текста. Результат вывести на экран в виде таблицы с графами буква – частота появления.
5.	Использовать шифрование с применением шифра Цезаря с величиной сдвига 4 по алфавиту в сторону убывания. Используется кольцевой сдвиг: буква А меняется на Ь, Б – на Э, В – на Ю, Г – на Я. Пробелы и знаки препинания не изменяются. Пример: До шифрования: МАМА МЫЛА РАМУ После шифрования: ИЬИЬ ИЧЗЬ МЬИП
6.	Задать размер строки (не менее самой длинной) и выровнять все строки по ширине (как это делается в Word) вставкой пробелов.
7.	Удалить из текста все гласные буквы.
8.	Переставить местами строки со сдвигом: первую на место второй, вторую на место третьей и т.д., последнюю на место первой.
9.	Использовать шифрование с применением шифра Цезаря и величиной сдвига 3 по алфавиту в сторону возрастания. Используется кольцевой сдвиг: буква Э меняется

	на А, Ю – на Б, Я – на В. Пробелы и знаки препинания не изменяются. Пример: До шифрования: МАМА МЫЛА РАМУ После шифрования: ПГПГ ПЮОГ УГПЦ
10.	Поменять слова со сдвигом на 2 слова: первое на место третьего, второе на место четвертого и т.д., предпоследнее на место первого, последнее на место второго.
11.	Продублировать все гласные буквы: везде, где встречается буква А записать АА, везде, где Е – ЕЕ и т.д.
12.	В каждой строке поменять местами первые и последние 3 буквы.
13.	Использовать шифрование методом циклической перестановки с длинной блока 4. Размер текста должен быть кратен 4, в противном случае текст дополняется пробелами. Текст по блокам сдвигается вправо, последняя буква помещается на место первой. Пример: До шифрования: |МАМА| МЫЛ|А РА|МУ | После шифрования: |АМАМ|Л МЫ|АА Р| МУ	
14.	Удалить из текста все знаки препинания и прочие специальные символы, оставив только пробелы. Каждую строку заканчивать точкой, если ее там не было.
15.	Поменять слова со сдвигом: первое на место второго и т.д., последнее на место первого.
16.	Продублировать буквы начала алфавита А, Б, В, Г, Д: везде, где встречается буква А записать АА, везде, где Б – ББ и т.д.
17.	Провести анализ вхождения символов-разделителей: пробелов, табуляции, перехода на новую строку. То есть определить отношение появления этих символов как ко всем буквам текста, так и ко всем символам файла.
18.	Использовать шифрование с применением шифра Цезаря и величиной сдвига 4 по алфавиту в сторону возрастания. Используется кольцевой сдвиг: буква Ь меняется на А, Э – на Б, Ю – на В, Я – на Г и т.д. Пробелы и знаки препинания не изменяются. Пример: До шифрования: МАМА МЫЛА РАМУ После шифрования: РДРД РЯПД ФДРЧ
19.	Поменять слова со сдвигом на 3 слова: первое на место четвертого, второе на место пятого и т.д., последнее на место третьего.
20.	Продублировать буквы начала и конца алфавита А, Б, В, Э, Ю, Я: везде, где встречается буква А записать АА,

	везде, где Б – ББ и т.д.
21.	Задать размер строки (не менее самой длинной) и выровнять все строки по ширине (как это делается в Word) вставкой пробелов.
22.	Удалить из текста все согласные буквы.
23.	Провести анализ вхождения знаков препинания: запятой, точки, точки с запятой, многоточия, восклицательного и вопросительного знаков. То есть определить отношение появления этих символов как ко всем буквам текста, так и ко всем символам файла.
24.	Использовать шифрование с применением шифра Цезаря с величиной сдвига 3 по алфавиту в сторону убывания. Используется кольцевой сдвиг: буква А меняется на Э, Б – на Ю, В – на Я. Пробелы и знаки препинания не изменяются. Пример: До шифрования: МАМА МЫЛА РАМУ После шифрования: ЙЭЙЭ ЙШИЭ НЭЙР
25.	Сделать анализ частоты появления букв в файле, упорядоченных по алфавиту. То есть результат – это таблица из 33 строк, в одной графе которой название буквы, в другой – количество появлений в файле.
26.	Подсчитать количество слов, пробелов и знаков препинания в файле. Результат с сопровождающими надписями вывести на экран. Записывать результат в файл не надо.
27.	Использовать шифрование методом циклической перестановки с длинной блока 3. Размер текста должен быть кратен 3, в противном случае текст дополняется пробелами. Текст по блокам сдвигается вправо, последняя буква помещается на место первой. Пример: До шифрования: |МАМ|А М|ЫЛА| РА|МУ | После шифрования: |АММ| МА|ЛАЫ|РА |У М|
28.	Отсортировать все слова по алфавиту, учитывая все буквы. Получить файл из строк по количеству слов, удалив все пробелы и знаки препинания.
29.	Провести анализ вхождения гласных А, Е, И, О, У, Э, Ю, Я. То есть определить отношение появления этих букв ко всем буквам текста.
30.	Использовать шифрование методом простой перестановки с длинной блока 4. Размер текста должен быть кратен 4, в противном случае текст дополняется пробелами. В данном случае местами меняются только край-

	ние буквы, средняя остается на месте. Пример: До шифрования: |МАМА| МЫЛ|А РА|МУ | После шифрования: |АМАМ|ЛЫМ |АР А| УМ|

Лабораторная работа № 12. Динамические переменные. Списки

В языке Паскаль, есть переменные, которые создаются и уничтожаются в процессе выполнения программы. Они не входят в явные описания программы и, следовательно, к ним нельзя обращаться с помощью имен. Память для них выделяется только динамически в ходе работы программы.

Доступ к динамическим переменным осуществляется с помощью указателей (или ссылок), которые становятся определенными после создания динамического объекта.

Динамические переменные очень широко используется в программировании, более того, современные операционные системы имеют функции поддержки динамических областей памяти.

Так как динамические переменные отсутствуют в разделах описаний, до для указания на них используются **указатели**. Тип указателя сам по себе не является динамической структурой данных, часто его называют **ссылочным типом**. Это обычное четырехбайтное (или больше, в зависимости от установленной разрядности и операционной системы) число, содержащее адрес, с которого начинается динамическая переменная. Адрес в архитектуре процессоров фирмы Intel – два целых беззнаковых числа, определяющие номер сегмента памяти и смещение внутри этого сегмента. Указатели используются для установления отношений или связей между динамическими структурами данных. Эти связи могут быть весьма сложными.

В стандартном языке Паскаль указатели должны ссылаться на однотипные элементы данных, то есть являются **типизированными**. Существует специальное значение указателя **NIL (пустой указатель)**, принадлежащее всем типам указателей. В этом случае указатель не указывает ни на какой элемент. Это значение применяется для обозначения конца списка или ветви дерева (по аналогии в функцией EOF).

Для того чтобы присвоить переменной ссылочного типа определенное значение, необходимо воспользоваться **операцией взятия адреса** (указателя), – символа амперсанд «@» и переменной базового типа.

Для реализации косвенного доступа к переменной через указатель используется **разыменование**. То есть, чтобы по

указателю получить доступ к переменной, необходимо после указателя поставить знак «^».

Основные действия над динамическими переменными — это их создание и уничтожение. Первое реализуется стандартной процедурой

NEW (<указатель>);

Для освобождения памяти, выделенной под динамическую переменную, используется процедура, обратная по действию процедуре New:

DISPOSE (<указатель>);

Работа с динамическими переменными требует большой аккуратности, иначе «засорение» памяти ненужными переменными может привести к быстрому ее переполнению.

Связный список – базовая динамическая структура данных в информатике, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну или две ссылки («связки») на следующий и/или предыдущий узел списка. Принципиальным преимуществом перед массивом является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

Маркеры (указатели). Чаще всего указывают на начало списка. Обозначаются либо специальным символом, либо выделяются цветом, либо отмечаются как-то иначе. Часто маркеры не поддаются удалению обычными средствами, – например, для их удаления вводится специальный элемент меню. Обычно маркер является первым элементом списка, и при возможности удаления удаляется весь список.

Типы списков.

Кольцевой. Следующий элемент для конца списка является начальным элементом (и следующий для первого элемента – последний в списке). При обычной организации списков при просмотре влево остановка происходит на первом элементе (никаких действий не предпринимается). При просмотре вправо – на последнем.

Двусвязный. Перемещаться по списку можно как от начала к концу, так и от конца к началу. Для односвязных списков движение возможно только в одну сторону.

Задание.

Использовать динамические переменные со структурой «Запись», состоящие из информационного поля заданного типа и одного или двух указателей.

Предусмотреть с выбором из меню 6 пунктов работы с динамическими переменными:

1. Создание списка (с формированием начальных условий)
2. Запись в файл
3. Удаление
4. Вставка
5. Просмотр вправо
6. Просмотр влево (для односвязных списков – только одно направление)
7. Чтение из файла (опционально)

Варианты заданий приведены в табл.13.

Таблица 13. Варианты заданий

№ вар.	Тип данных	Тип списка
1	Целый	Кольцевой, двусвязный, с удалением маркера из специального пункта меню.
2	Вещественный	Двусвязный, с неудаляемым маркером начала списка
3	Логический	Кольцевой, движение вправо с удалением маркера из специального пункта меню.
4	Строка	Кольцевой, двусвязный, с удалением маркера из специального пункта меню.
5	Целый	Двусвязный, с удалением маркера из специального пункта меню
6	Вещественный	Кольцевой, движение влево с удалением маркера из специального пункта меню.
7	Логический	Кольцевой, двусвязный, с удалением маркера из специального пункта меню.
8	Строка	Двусвязный, с неудаляемым маркером начала списка
9	Целый	Кольцевой, движение вправо с удалением маркера из специального пункта меню.
10	Вещественный	Кольцевой, двусвязный, с удалением маркера из специального пункта меню.
11	Логический	Двусвязный, с удалением маркера из специального пункта меню
12	Строка	Кольцевой, движение влево с удалением маркера из специального пункта меню.
13	Целый	Кольцевой, двусвязный, с удалением маркера из специального пункта меню.
14	Вещественный	Двусвязный, с неудаляемым маркером начала списка
15	Логический	Кольцевой, движение вправо с удалением маркера из специального пункта меню.
№	Тип данных	Тип списка

вар.		
16	Строка	Кольцевой, двусвязный, с удалением маркера из специального пункта меню.
17	Целый	Двусвязный, с удалением маркера из специального пункта меню
18	Вещественный	Кольцевой, движение влево с удалением маркера из специального пункта меню.
19	Логический	Кольцевой, двусвязный, с удалением маркера из специального пункта меню.
20	Строка	Кольцевой, движение вправо с удалением маркера из специального пункта меню
21	Целый	Двусвязный, с неудаляемым маркером начала списка
22	Вещественный	Двусвязный, с удалением маркера из специального пункта меню
23	Логический	Кольцевой, движение влево с удалением маркера из специального пункта меню.
24	Строка	Кольцевой, двусвязный, с удалением маркера из специального пункта меню.
25	Вещественный	Кольцевой, движение вправо, с неудаляемым маркером начала списка
26	Логический	Кольцевой, двусвязный, с неудаляемым маркером начала списка
27	Целый	Кольцевой, движение вправо с удалением маркера из специального пункта меню.
28	Строка	Кольцевой, двусвязный, с удалением маркера из специального пункта меню.
29	Вещественный	Двусвязный, с неудаляемым маркером начала списка
30	Логический	Двусвязный, с удалением маркера из специального пункта меню

Лабораторная работа № 13. Графический режим монитора. Построение графиков

Целью работы является ознакомление с использованием функций графического режима работы монитора на примере построения графиков.

Так как Турбо-Паскаль разрабатывался под ОС MS DOS, то основным режимом работы монитора был текстовый. С появлением Windows 95 стал графический режим. Поэтому Турбо-Паскаль (так же, как и Free Pascal) ориентирован на работу в текстовом режиме. Так как модуль Graph в Турбо-Паскале и Free Pascal существенно различаются, то рассмотрим только модуль для Free Pascal.

Для перевода в графический режим необходимо выполнять определенные действия.

1. Подключить модуль Graph, в котором находятся все процедуры и функции графического режима монитора. Делается это с помощью описания подключаемых модулей:

```
Uses Graph
```

   Это описание следует сразу же за заголовком программы.
2. Для перехода в графический режим (на месте блока в алгоритме «Очистка экрана») могут использоваться процедуры DetectGraph и InitGraph, например:

```
DetectGraph (driver,razmer);
InitGraph (driver,razmer,'');
```

   Здесь driver задает номер драйвера видеорежима, razmer – размер экрана в пикселях, оба параметра – целые. Последний параметр в InitGraph (пустая строка) задает шрифт по умолчанию. Эта пустая строка пишется всегда, так как дополнительные шрифты с Free Pascal не поставляются.
3. Затем задаются цвета фона и переднего плана (точек, линий, букв и т.д.), по умолчанию они соответственно черный и белый. Так как выводятся два графика, то их лучше всего обозначать разными цветами. Для установки цвета переднего плана используется процедура SetColor с одним целым параметром, который указывает цвет:

   0 – черный;

1 – синий;
2 – зеленый;
3 – светло-синий;
4 – красный;
5 – фиолетовый;
6 – коричневый;
7 – светло-серый;
8 – серый;
9 – светло-синий;
10 – светло-зеленый;
11 – голубой;
12 – светло-коричневый;
13 – светло-фиолетовый;
14 – желтый;
15 – белый.

4. В модуле Graph много различных процедур и функций, предназначенных для рисования геометрических фигур. Кроме этого, координаты каждой фигуры могут быть как абсолютными, так и относительными. В этой лабораторной работе используются только абсолютные координаты и несколько процедур, например, такие:

```
OutTextXY(X0,Y0,Text)
```

Здесь X0 – смещение по оси X с начала координат,
Y0 – смещение соответственно по Y, обе целые,
Text – выводимый, начиная с указанной точки, текст.

Начало координат находится в **левом верхнем углу**, то есть увеличение значений по Y приводит не к подъему по оси, а к спуску.

```
Line(X0,Y0,X1,Y1)
```

Прямая линия с начальными координатами X0 и Y0, и конечными X1 и Y1.

```
PutPixel(X0,Y0,color)
```

Вывод точки (пикселя) на экран с указанными координатами и цветом.

Для Pascal ABC подключаемый модуль называется GraphABC. Кроме этого, пропускаются пункты 2 и 3.

Вместо процедуры OutTextXY используется TextOut с теми же параметрами. Процедура Line изменений не претерпела, процедура PutPixel заменена на SetPixel с теми же параметрами, кроме цвета. Цвет выбирается из списка предопределенных имен, описанных в справке по Pascal ABC: «Стандартные модули» > «Модуль GraphABC» > «Цветовые константы». Кроме этого, цвет линий и текста может быть установлен процедурой SetPenColor.

Требования к оформлению графиков функций.

Чтобы график не был, с одной стороны, с очень маленьким разрешением по высоте, с другой не вылезал за экран, необходимо функцию отмасштабировать. То есть сначала в программе вычисляются минимальное и максимальное значения, затем вручную они округляются в бо́льшую по модулю сторону. Так как округление может производиться не обязательно по целым числам, то проще это сделать вручную, как описано в блок-схеме алгоритма к этой работе (рис.15.1).

График должен иметь заголовок и шкалы с указанием осей, засечек на шкалах. Обозначение осей абсцисс и ординат и проставленными по ним значениями. То есть программа должна иметь, кроме общего цикла вывода графиков, ещё 2 цикла: по осям X и Y.

Варианты заданий приведены в табл.14.



Рис. 31. Общая блок-схема алгоритма вывода графиков функций.

Таблица 14. Варианты заданий

№ вар.	Функции	Начальное значение x	Конечное значение x
1	$\ln x$ и $\lg x$	1	20
2	$\sin e^{-x}$ и $\cos e^{-x}$	0,05	1
3	$\arcsin e^{-x}$ и $\arccos 10^{-x}$	2	5
4	$\sqrt[3]{x}$ и $\sqrt[5]{x}$	20	100
5	e^{-x} и 10^{-x}	1	2
6	$\mathrm{tg}^3 x$ и $\mathrm{ctg}^3 x$	0,6	1,5
7	$e^{x}\sin x$ и $e^{-x}\sin x$	0	1
8	$\sqrt{\ln x}$ и $\sqrt[4]{\ln x}$	1	5
9	$\cos\sqrt{x}$ и $\mathrm{tg}\sqrt{x}$	0,2	2
10	10^{-x} и $10^{\sqrt{x}}$	0,1	2
11	$\arcsin \frac{1}{x^2}$ и $\arccos \frac{1}{x^3}$	1,05	2
12	$\ln\frac{1}{x}$ и $\lg\frac{1}{x}$	0,1	1
13	$\sin\frac{1}{\sqrt{x}}$ и $\cos\frac{1}{\sqrt{x}}$	0,2	2
14	$x^{0,3(\pi+1)}$ и $x^{(e-1)}$	0,5	4
15	$e^{x}\mathrm{tg}\, x$ и $e^{-x}\mathrm{ctg}\, x$	0,05	1

№ вар.	Функции	Начальное значение x	Конечное значение x
16	$\sqrt[5]{\cos x^2}$ и $\sqrt[5]{\sin x^3}$	0,05	1
17	$e^{\sqrt[3]{x}}$ и $e^{\sqrt[5]{x}}$	1	10
18	$10^{\operatorname{tg} x}$ и $10^{\sin x}$	0,25	0,5
19	$(x+\pi)^{0,15}$ и $(x+\pi)^{0,35}$	-1	1
20	$\ln\sqrt{x+\pi}$ и $\lg\sqrt[3]{x+\pi}$	-2	2
21	$\sqrt[15]{x+e}$ и $\sqrt[25]{x+e}$	0	100
22	$\operatorname{sh}\sqrt{x}$ и $\operatorname{ch}\sqrt{x}$	1	5
23	$\frac{1}{\sqrt{\operatorname{sh} x}}$ и $\frac{1}{\sqrt{\operatorname{ch} x}}$	1	2
24	$\operatorname{tg}\sqrt{\sin x}$ и $\operatorname{ctg}\sqrt{\cos x}$	0	1
25	$x^3+\operatorname{tg} x$ и $x^2+\operatorname{ctg} x$	0,1	1
26	$\sqrt[3]{x^2+\sin x}$ и $\sqrt[3]{x^2+\cos x}$	0	2
27	$e^{-x}+x^{0,6}$ и $10^{-x}+x^{0,7}$	1	3
28	$\ln\operatorname{tg}\sqrt{x}$ и $\lg\operatorname{ctg}\sqrt{x}$	0,1	1,5
29	$\operatorname{sh}(x^2+1)$ и $\operatorname{ch}(x^3-1)$	0	2
30	$\operatorname{tg}\sqrt{x^2-2}$ и $\operatorname{ctg}\sqrt{x^2+2}$	3	10

Приложение А. Основные стандартные функции

В приложении А собраны основные функции модулей, подключаемых автоматически (для Турбо-Паскаля и Free Pascal это модуль System, для Pascal ABC.NET – модуль PABCSystem), то есть используемые по умолчанию.

Для Free Pascal используется дополнительный модуль «Math», который нужно подключать принудительно. Функции, входящие в модуль «Math», отмечены «*».

Таблица 15. Стандартные функции ИСР Турбо-Паскаль, Free Pascal и Pascal ABC.NET, версия 2.2

Название функции	Математическая запись	Запись на языке Паскаль	Наличие в Турбо-Паскале	Наличие в Free Pascal	Наличие в Pascal ABC
Математические функции					
Абсолютная величина	|x|	Abs(x)	есть	есть	есть
Арктангенс	arctg x	Arctan(x)	есть	есть	есть
Косинус	cos x	Cos(x)	есть	есть	есть
Экспоненциальное значение	e^x	Exp(x)	есть	есть	есть
Дробная часть числа, результат - вещественный		Frac(x)	есть	есть	есть
Старший байт аргумента		Hi(i)	есть	есть	нет
Целая часть числа, результат - вещественный		Int(x)	есть	есть	есть
Натуральный логарифм	ln x	Ln(x)	есть	есть	есть
Младший байт аргумента		Lo(x)	есть	есть	нет
Является ли аргумент нечетным числом?		Odd(i)	есть	есть	есть
Значение числа π	π	Pi	есть	есть	есть
Случайное число в диапазоне 0-i		Random(i)	есть	есть	есть
Случайное вещественное число		Random	есть	есть	есть
Округление по правилам арифметики, результат - длинное целое		Round(x)	есть	есть	есть
Синус	sin x	Sin(x)	есть	есть	есть

Квадрат аргумента	x^2	Sqr(x)	есть	есть	есть
Квадратный корень	$\sqrt{x}$	Sqrt(x)	есть	есть	есть
Переставляет местами старший и младший байты		Swap(i)	есть	есть	есть(1)
Округление значения с отбрасыванием дробной части		Trunc(x)	есть	есть	есть
Арккосинус	arccos x	Arccos(x)	нет	есть*	есть
Арккосинус гиперболический	arcch x	Arccosh(x) Arcosh(x)	нет	есть*	нет
Арксинус	arcsin x	Arcsin(x)	нет	есть*	есть
Арксинус гиперболический	arcsh x	Arcsinh(x) Arsinh(x)	нет	есть*	нет
Арктангенс с аргументом x/y	arctg x/y	Arctan2(x,y)	нет	есть*	нет
Арктангенс гиперболический	arcth x	Arctanh(x) Artanh(x)	нет	есть*	нет
Округление в направлении к положительной бесконечности		Ceil(x)	нет	есть*	есть
Косеканс	cosec x	Cosecant(x) Csc(x)	нет	есть*	нет
Косинус гиперболический	ch x	Cosh(x)	нет	есть*	нет
Котангенс	ctg x	Cot(x) Cotan(x)	нет	есть*	нет
Преобразование оборотов в радианы		CycleToRad(x)	нет	есть*	нет
Преобразование градусов в грады		DegToGrad(x)	нет	есть*	нет
Преобразование градусов в радианы		DegToRad(x)	нет	есть*	есть
Округление в направлении к отрицательной бесконечности		Floor(x)	нет	есть*	есть
Преобразование градов в градусы		GradToDeg(x)	нет	есть*	нет
Преобразование градов в радианы		GradToRad(x)	нет	есть*	нет
Возведение x в целую степень i	x^i	Intpower(x,i)	нет	есть*	нет
Вычисление экспоненциальной функции	$x*2^i$	Ldexp(x,i)	нет	есть*	нет
Вычисление натурального логарифма	$\ln(x+1)$	LnXP1(x)	нет	есть*	нет
Вычисление десятичного логарифма	$\log x$	Log10(x)	нет	есть*	есть
Вычисление двоичного логарифма	$\log_2 x$	Log2(x)	нет	есть*	есть
Вычисление логарифма $x2$ по основа-	Log_{x1}	Logn(x1,x2)	нет	есть*	есть

нию **x**1	*x*2				
Максимальное их двух значений		Max(a,b)	нет	есть*	есть
Максимальное значение в целочисленном массиве		MaxIntValue(a)	нет	есть*	нет
Максимальное значение в целочисленном или вещественном массиве		MaxValue(a)	нет	есть*	нет
Минимальное их двух значений		Min(a,b)	нет	есть*	есть
Минимальное значение в целочисленном массиве		MinIntValue(a)	нет	есть*	нет
Минимальное значение в целочисленном или вещественном массиве		MinValue(a)	нет	есть*	нет
Возведение *x*1 в дробную степень *x*2	$x1^{x2}$	Power (x1,x2)	нет	есть*	есть
Преобразование радианов в обороты		RadToCycle(x)	нет	есть*	нет
Преобразование радианов в градусы		RadToDeg(x)	нет	есть*	есть
Преобразование радианов в грады		RadToGrad (x)	нет	есть*	нет
Секанс	sec *x*	Secant(x) Sec(x)	нет	есть*	нет
Синус гиперболический	sh *x*	Sinh(x)	нет	есть*	нет
Тангенс гиперболический	th *x*	Tanh(x)	нет	есть*	нет
Функции для работы со строками и символами					
Преобразование целого числа *i* в символы, содержащие *k* двоичных разрядов		BinStr(i,k)	нет	есть	нет
Символ, соответствующий указанному значению кода ASCII		Chr(i)	есть	есть	есть
Конкатенация (объединение) последовательности строк		Concat(s1, …) или +	есть	есть	есть
Копирование подстроки с количеством символов i2 из строки s, начиная с i1		Copy(s,i1, i2)	есть	есть	есть
Преобразование целого значения *i* в символы, содержащие k шестнадцатеричных разрядов		HexStr(i,k)	нет	есть	нет
Динамическая длина строки		Length(s)	есть	есть	есть
Преобразует символы в строчные буквы		LowerCase(s)	нет	есть	нет
Преобразование целого значения *i* в символы, содержащие k восьмеричных разрядов		OctStr(i,k)	нет	есть	нет

Поиск подстроки s1 в строке s2, результат – целое (byte)		Pos(s1,s2)	есть	есть	есть
Задает содержимое строки и ее длину		SetString(s,c,i)	нет	есть	нет
Преобразование числового значения *x* в строковую переменную *s*		Str(x,s)	есть	есть	есть
Размножение символа *c* *i* раз с получением в результате строки		StringOfChar (c,i)	нет	есть	нет
Преобразует символы в прописные буквы		UpCase(s)	есть	есть	есть
Преобразование из строки *s* в числовую переменную *x*. Где *i* – признак завершения операции		Val(s,x,i)	есть	есть	есть
Функции для работы с памятью, адресами и указателями					
Адрес заданного объекта, результат - указатель		Addr(x)	есть	есть	нет
Проверка, имеет ли указатель или переменная значение nil		Assigned(p)	есть	есть	нет
Текущее значение регистра CS		CSeg	есть	есть	нет
Текущее значение регистра DS		DSeg	есть	есть	нет
Размер наибольшего непрерывного свободного блока в динамически распределяемой области		MaxAvail	есть	нет	нет
Число свободных блоков в динамически распределяемой области памяти		MemAvail	есть	нет	нет
Смещение в памяти заданной переменной		Ofs(x)	есть	есть	нет
Преобразует адрес, заданный в виде базового сегмента i1 и смещения i2, в значение типа указатель		Ptr(i1,i2)	есть	есть	нет
Сегмент для указанной переменной		Seg(x)	есть	есть	нет
Текущее значение регистра SP (указателя стека)		SPtr	есть	есть	нет
Текущее значение регистра SS		SSeg	есть	есть	нет
Функции для работы с файлами					
Конец файла		Eof(f)	есть	есть	есть
Конец строки		Eoln(f)	есть	есть	есть
Определяется текущая позиция в файле		FilePos(f)	есть	есть	есть
Определяется текущий размер файла		FileSize(f)	есть	есть	есть

Текстовый файл с именем f, открытый на чтение в кодировке Windows		OpenRead(f)	нет	нет	есть
текстовый файл с именем f, открытый на запись в кодировке Windows		OpenWrite(f)	нет	нет	есть
Определение для файла статуса «конец файла»		SeekEof(f)	есть	есть	есть
Определение для файла статуса «конец строки»		SeekEoln(f)	есть	есть	есть
Прочие функции					
Определяется максимум заданной величины		High(i)	есть	есть	есть
Целое значение, представляющее собой состояние последней выполненной операции ввода-вывода.		IOResult	есть	есть	нет
Определяется минимум заданной величины		Low(x)	есть	есть	есть[2]
Порядковый номер для значения перечислимого типа		Ord(x)	есть	есть	есть[3]
Число параметров, переданных в командной строке		ParamCount	есть	есть	нет
Заданный параметр командной строки		ParamStr(i)	есть	есть	есть
Предшествующее значение аргумента		Pred(x)	есть	есть	есть
Число байтов, занимаемых аргументом		SizeOf(x)	есть	есть	нет
Следующее значение агрумента		Succ(x)	есть	есть	есть
Указатель на таблицу методов объекта		TypeOf(x)	есть	есть	нет

Примечания. (1) В Pascal ABC функция Swap меняет местами не байты в одной переменной, а две переменных, поэтому имеет 2 параметра.
(2) В Pascal ABC функция Low возвращает 0.
(3) В Pascal ABC функция Ord преобразует символ в код в кодировке Unicode.

Приложение Б. Отличия языка ИСР PascalABC.NET от Delphi

Добавлено

1. Операции += -= для событий .NET и для процедурных переменных.
2. Операции += -= *= для целых и += -= *= /= для вещественных.
3. Операция += для строк.
4. Подпрограммы с переменным числом параметров.
5. Операция new для вызова конструктора (ident := new type_name(params);).
6. Операция new для создания динамического массива.
7. Операция typeof .
8. Использование uses для подключения пространств имен .NET (реализовано в Delphi Prism).
9. Вид доступа internal (наряду с public, private, protected).
10. Инициализация переменных: var a: integer := 1;
11. Инициализация переменных: var a := 1;
12. Объявление локальных переменных в блоке.
13. Объявление параметра цикла в заголовке цикла: for var i := 1 to 10 do, foreach x: real in a do.
14. Оператор lock, обеспечивающий синхронизацию потоков.
15. Методы в записях.
16. Инициализаторы полей в классах и записях.
17. Обобщенные классы (generics).
18. Реализованы типизированные файлы (в отличие от Delphi Prism, где они убраны).
19. Упрощенный синтаксис модулей.
20. Описание методов внутри интерфейса класса или записи.
21. Реализация записью интерфейса.
22. Методы расширения.
23. Лямбда-выражения.

Изменено

1. Только сокращенное вычисление логических выражений.
2. Другой синтаксис foreach.
3. Интерфейсы interface в стиле .NET.
4. Другой синтаксис перегрузки операций.

5. Статические методы классов вместо классовых методов. Отсутствие типа TClass.
6. Деструкторы оставлены лишь для совместимости и не выполняют никаких действий.
7. Тип object - синоним System.Object.
8. Тип exception - синоним System.Exception.
9. Индексация string с 1, директива переключения на индексацию с 0.
10. Процедура write выводит любые типы.
11. Структурная эквивалентность типов для процедурных переменных, динамических массивов, типизированных указателей и множеств (в Delphi Object Pascal - именная эквивалентность типов за исключением открытых массивов).
12. Множества на базе произвольных типов (set of string).
13. Запрет использования указателей на управляемую память.
14. Процедурные переменные (делегаты) вместо procedure of object.
15. С бестиповыми файлами file можно работать с помощью процедур read, write.
16. Массивы массивов отличаются по типу от двумерных массивов (в частности, записи a[i][j] и a[i,j] неэквивалентны).
17. Перегрузка выполняется без ключевого слова overload.
18. Все конструкторы имеют имя Create.
19. Автоматическое управление памятью с помощью сборщика мусора (за исключением указателей на неуправляемую память).

Отсутствует

1. Ключевые слова и директивы, и связанные с ними возможности:
    1. Packed
    2. Threadvar
    3. Inline
    4. Asm
    5. Exports
    6. Library
    7. Unsafe
    8. Resourcestring
    9. Dispinterface
    10. In
    11. Out
    12. Absolute

13. Dynamic
14. Local
15. Platform
16. Requires
17. Abstract
18. Export
19. Message
20. Resident
21. Assembler
22. Safecall
23. Automated
24. Far
25. Near
26. Stdcall
27. Cdecl
28. Published
29. Stored
30. Contains
31. Implements
32. Varargs
33. Default
34. Deprecated
35. Package
36. Register
37. Dispid
38. Pascal
39. writeonly.

2. Приведение типов для переменных: Char(b) := 'd'.
3. Возможность присвоить адрес подпрограммы указателю pointer.
4. Записи с вариантами.
5. Строки PChar.
6. Возможность использовать операцию @ для процедурных переменных.
7. Вариантные типы.
8. Бестиповые параметры (var a; const b).
9. Открытые массивы (не путать с динамическими!).
10. Методы, связанные с сообщениями (message).
11. Классовые свойства.
12. Вложенные определения классов.
13. Константы-поля классов.

Литература

1. Сайт «PascalABC.NET. Современное программирование на языке Pascal», http://pascalabc.net/
2. Цветков А.С. Язык программирования PASCAL. Система программирования ABC Pascal. Учебное пособие для школьников 7-9 классов. – СПб.: Павловск: Школа им. А.М. Горчакова, 2012-2013. – 46 с., ил.
3. Махно В.В., Михалкович С.С., Пучкин М.В. Основы программирования графики. Методические указания для преподавателей факультета математики, механики и компьютерных наук, ведущих курсы по основам программирования. – Ростов-на-Дону: Федеральное агентство по образованию, «Южный федеральный университет», 2007. – 49 с.
4. Сайт «Free Pascal. Open source compiler for Pascal and Object Pascal», http://www.freepascal.org/
5. Алексеев Е.Р., Чеснокова О.В., Кучер Т.В. Free Pascal и Lazarus: Учебник по программированию. – М.: ALT Linux; Издательский дом ДМК-пресс, 2010. – 440 с.: ил. – (Библиотека ALT Linux).
6. Кетков Ю.Л. Свободное программное обеспечение. FREE PASCAL для студентов и школьников. – СПб.: БХВ-Петербург, 2011. — 384 с.: ил.
7. Павловская Т.А. Паскаль. Программирование на языке высокого уровня. Учебник для вузов. – СПб.: Питер, 2007. – 393 с.: ил.
8. Турбо Паскаль 7.0. Самоучитель. – СПб.: Питер; К.: Издательская группа BHV, 2002. – 416 с.: ил.

Полетаев Игорь Алексеевич
Полетаева Ольга Александровна

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ ПАСКАЛЬ

Методические указания

по выполнению лабораторных работ для студентов очной формы обучения направлений подготовки
09.03.01 «Информатика и вычислительная техника»,
09.03.02 «Информационные системы и технологии»,
09.03.04 «Программная инженерия»,
13.03.02 «Электроэнергетика и электротехника»

Технический редактор Полетаев И.А.
Компьютерная верстка Полетаев И.А.

Отпечатано с готового оригинал-макета,
предоставленного авторами

Подписано в печать ____________ Формат 60x90/16.
Гарнитура Bookman. Усл. печ. л. 4,75
Тираж _____ экз. Заказ № _____

Адрес издательства:
Россия, 180000, Псков, ул. Л.Толстого, 4
Издательство ПсковГУ